C

MPASS

By
Jason M. Goertz
& Jerry Fochtman

The definitive guide to MPE-V low-level procedures

COMPASS
The Definitive Source of Information on MPE-V Low Level Procedures

Jerry Fochtman
Jason M. Goertz

Propellerhead Publishing
Sumner, WA 98390

In Association With
CO-OP Publishing
Austin, TX 78759

©Propellerhead Publishing, Mattedor Computer Services and Jerry Fochtman, 1989.

All rights reserved. No part of this book may be reproduced, stored or transmitted in any form by any means electronic, mechanical, photocopying, recording, or otherwise, without prior written permission of the publisher. Published in the United States of America.

This book was typeset in Bitstream Roman, Roman Bold, Roman Italic, Swiss, Swiss Bold, and Hewlett-Packard Letter Gothic Type Fonts. Original pages were typeset on a Compuadd 286 computer using WordPerfect 5.0 to a LaserJet Series II under MS-DOS 3.3.

MPE-V is a product of Hewlett-Packard.

NOTICE

The information contained in this document is subject to change without notice.

PROPELLERHEAD PUBLISHING, JERRY FOCHTMAN, and JASON GOERTZ MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Propellerhead Publishing shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

Propellerhead Publishing assumes no responsibility for the use or reliability of the information contained herein.

This document contains proprietary information which is protected by copyright. All rights are reserved. No part of this document may be photocopied, reproduced or translated to another language without prior written consent of Propellerhead Publishing.

LIST OF EFFECTIVE PAGES

The List of Effective Pages gives the date of the current edition, and lists the dates of all changed pages. Unchanged pages are listed as "ORIGINAL". Within the manual, any page changed since the last edition is indicated by printing the date the changes were made on the bottom of the page. Changes are marked with a vertical bar in the margin. If an update is incorporated when an edition is reprinted, these bars and dates remain. No information is incorporated into a reprinting unless it appears as a prior update.

First Edition . . . . . September, 1989

PRINTING HISTORY

New editions are complete revisions of the manual. Update packages, which are issued between editions, contain additional and replacement pages to be merged into the manual by the customer. The date on the title page and back cover of the manual changes only when a new edition is published. When an edition is reprinted, all the prior updates to the edition are incorporated. No information is incorporated into a reprinting unless it appears as a prior update.

First Edition . . . . . Aug, 1989

Acknowledgements and Dedication

The authors wish to thank the following individuals for their direct or indirect contributions to this work:

Doris Goertz and Donna Fochtman - For their tireless patience when the authors "had to figure out one more procedure", taking them away from house remodeling, taking their turn with the kids, etc.

Ron Seybold (HP Chronicle) - For all his encouragement and advice, and his ability to elicit good columns month after month.

Stan Sieler (Allegro Consultants) - For his editing, advice, and admonitions about the esoterica of the procedures documented here.

This work is dedicated to some pioneers in the world of MPE-V internals, whose shoulders we stand on in presenting this volume of information:

Gerry Wade (HP) - Former HP SE, now at HP Roseville, author of multitudes of early MPE utilities, such as LISTFXX, TAPESCAN, LIST and probably OVERLORD and a version of SOO or three.

Ed McCracken - Former head of HP's General Systems Division (maker of MPE and the HP 3000). Without his presence at the HP User's Group Meeting in Denver in 1978, we probably would still not have a Systems Tables Manual.

Bernie Staley (HP) - HP TSE in HP's Ridgeton, Missouri (Saint Louis) office. Another pioneer in the world of MPE utilities. Author of BULDACCT, GETFILE and gazillion other programs which were used by many as models for Priv Mode programming in SPL.

MPE Lab - To the many people who, over last 18-odd years, have worked for HP and written, maintained, and otherwise contributed to the MPE-V operating system. Without these folks, MPE wouldn't exist, and there would be no need for COMPASS at all.

Eugene Voloch - Author of MPEX, widely regarded as the quintessential PM program. A youngster, yes, but wise beyond his years in the intricacies of MPE internals, and one who has not been afraid to share his knowledge with others, the founding principle behind COMPASS.

We realize that there have been many, many others through the years who have contributed to the knowledge base of MPE internals. To those, we offer a blanket dedication and thanks for advancing the state-of-the-art in MPE.

Table of Contents

Section I - Introduction

This Book's Audience 1
How To Use This Manual 2

Section II - Procedures

Introduction and Conventions 1
Abbreviations 1
Procedure Options 3
ABORT 1
ABORTIO 4
ABORTJOB 5
ABORTPROCIO 6
ABORTTIMEREQ 7
ACCCHECK 8
ADDJTENTRY 10
ADOPT 12
ALLORIN 13
ALTDSEGSIZE 15
ALTEROUTPUTFILE 16
ATTACHIO 18
AWAKE 22
BUILDSTACKMARKER 24
BURRYPROC 25
C'CLOCK 26
C'DATE 27
CACHE'LDEV 28
CDT'SET'POST 29
CDT'SET'RND 30
CDT'SET'SEQ 31
CENTER 32
CHANGEDB 33
CHECK'BOT 35
CHECKALIVE 36
CHECKALLOW 37
CHECKDISC 39
CHECKLDEV 41
CHEK 42
CLEANUPFILES 45
CLEAR'PSIF 46
CLEARWWS 48
CLEAR'BREAKPNT 49
CONSALTFILE 52
CONVERTDATE 53
CONVERTTIME 54
CONVEXTLABELTODELTAP 55
COMSEARCH 56
CRYPT 58
DATEJUL 59
DEALLORIN 60
DEBLANK 61

Table of Contents

DELAY 62
DELETEDEVFILE 63
DELINKSCHED 65
DEVSPEC 66
DIRECADJUST 67
DIRECFIND 69
DIRECFINDFILE 72
DIRECINSERT 75
DIRECINSERTFILE 78
DIRECLOGOFF 81
DIRECLOGON 83
DIRECPURGE 85
DIRECPURGEFILE 88
DIRECSCAN 90
DIREC[RE]SETFLAG 94
DIRMATCH 97
DISCSIZE 98
DISCTYPE 99
DISMOUNT 100
DSMOVE 102
DSOPEN 103
DST'SIZE 104
ERROREXIT 105
ERRORON 107
EXCHANGEDB 108
FALTSEC 109
FAMILY 111
FGETLOCKWORD 112
FINDDEVICES 113
FLABIO 115
FMAKETEMP 117
FORMSG 118
FORMUSERID 122
FREEZESEG' 123
FREPLY 125
FROMCALENDAR 126
FSCLOSE 127
FSECTORS 128
FSOPEN 129
GENMSG 131
GENMSGU 135
GET'DSDEVICE 137
GET'FILECODE 139
GET'FILEMNEMONI 140
GET'SYS'PRIV 141
GETCLASS 142
GETCLOCK 144
GETDATASEG 146
GETDEVINFO 148
GETJITDST 150
GETJOBN 151

Table of Contents

GETMAXJS 152
GETPCBXMEASINFO 153
GETSALT 155
GETSECINFO 157
GETSIR 158
GETSTATISTICS 161
GIVECLOCK 163
HELPPROC 164
IDENTIFY'TERMIN 165
IMPEDE 167
IOCONTROL 168
IOFREEZE 169
IOFREEZE' 171
IOSTAT 173
IOUNFREEZE 175
IOUNFREEZE' 177
ISSYSCACHABLE 179
JULDATE 180
LDEVTOTYPE 181
LDEVTOVTAB 182
LOCKJIR 183
LOCKSEG 184
LOCKSEG' 186
LOGICALCST 188
LUN 189
MAKEFN 190
MASTEROP 191
MAX'DST'SIZE 192
MOUNT 193
MPE 195
MUSTOPEN 196
MYPIN 197
OBTAIN 198
PCB'INDEX 200
PHYSICALCST 201
PINJOBMAP 203
PROCESS'CODE'LOAD'INFO 204
PROCFILE 206
PRODUCEPARMS 207
QUEUEPROC 209
RELDATASEG 210
RELEASE 211
RELSIR 212
REMOVE'PASSWORDS 213
REQUESTSERVICE 214
RESETBREAKBITS 215
RESETCRITICAL 216
RESETDB 217
RLOCK 218
RUNLOCK 219
SCHEDDAYTIME 220

Table of Contents

SCHEDULESCHED 221
SECHK'PASSWORD 222
SECURITYINSTALLED 224
SEGMENTER 225
SET'BREAKPNT 229
SET'PSIF 231
SETCRITICAL 232
SETDBTOSTACK 233
SETSYSDB 234
SFINDODD 235
SHUTDOWNCACHING 236
STACK'DST 237
STACKCHECK 238
STARTSTATISTICS 239
STOPSTATISTICS 241
STRINGPINATTAIL 242
SYSGLOB 243
SYSPROC 244
THISCPU 245
TIMEREQ 247
TOCALENDAR 249
UDC 250
UNCACHE'LDEV 251
UNFREEZESEG' 252
UNIMPEDE 254
UNLOCKJIR 255
UNLOCKSEG 256
UNLOCKSEG' 258
UNSTRINGHEADPIN 260
VERIFY'PASSWORD 261
VTABINX 263
VTABTOLDEV 264
WAIT 265
WHERES'DB 267
XCODETRAP 268
XRETJTENTRY 269

Section III - Expansions
Introduction 1
Directory 1
Theory 1
Practice 2
File Entries 3
Private Volumes 3
Examples 3
Finding 4
Purging 5
Inserting 8
Scanning 11
DIRECSCAN Parms 11
The RECIP Procedure 12

Table of Contents

A Working Program 13
Conclusion 22
IO System 23
ATTACHIO 23
ATTACHIO Parameters 23
IOCDPN0 24
Charts 25
Functional Returns 25
Disc Operations 26
Mag Tape 26
Line Printer 26
Terminals 26
Conclusion 28
MPE-V Security 29
Description 29
New Security Procedures 29
Locating Commands 29
Device Password Encryption 29
System Security Table 29

About the Authors

Index

Reader Comment Form

Section I - Introduction

Welcome to COMPASS, the Definitive Source of Information on MPE-V Low Level Procedures. It is our hope that you can derive a great deal of use from this manual. We also hope that you share our passion for this useful operating system, MPE.

Both authors have been playing with computers for many years, and with the internals of MPE for most of the time MPE has been around. We now wish to share with you the wealth of information that we have amassed, up to now for our own use.

COMPASS is divided into two main parts. The first, Section II, is where the bulk of the information presented lies. In this Section, the actual MPE procedures are documented. The use, calling sequence, parameter definitions, and other considerations are presented in some detail. It is here that you will probably spend most of your time.

The second main part, Section III, we call Expansions. Here, we expand on the discussions presented in Section II for a few useful classes of procedures: the Directory, IO, and Security.

This Book's Audience

Along the way, we have assumed that the reader of this book will be knowledgeable in MPE-V internals, or at least a very proficient SPL programmer looking to do some more in-depth work with MPE. This book is **not** a course on MPE-V internals, although the reader may pick up a thing or two along the way. It is primarily intended to help the working programmer to write better, tighter, and more functional code.

We don't advocate using the procedures in this manual if another, more supported, and safer method exists. To create an account, for example, one could call DIRECINSERT, but wouldn't calling the COMMAND intrinsic (a documented, supported routine) and issuing a :NEWACCT command be safer? But sometimes, using the low level procedure is a better way to go, simply because the procedures are faster, or more functional, or bypass some edit checking (or security checking) that slows things down.

Along with this speed and convenience, however, comes responsibility. **No one should expect to use these procedures without possibly paying a price at some point, perhaps a system failure.** The procedures in COMPASS are faster and more convenient in many ways because they are "trusted" interfaces. That is, they "trust" the caller to provide correct data and parameters. System intrinsics perform all sorts of edit checks on the parameters, and abort or return errors if the parameters are not correct. Many of the procedures in here do not do any such edit checking, and can cause problems with your system if the data provided is not correct. Therefore, we must provide the following disclaimer:

> Neither Propellerhead Publishing, Jerry Fochtman, Jason Goertz, or Hewlett-Packard warrant the use of the procedures in this book. **Use them at your own risk.** Every attempt has been made to provide correct and current information regarding these procedures, but they can change at any time. To ensure your system integrity, use these procedures carefully and verify the parameters passed carefully while in testing.

Don't let the previous statement scare you too much. It has to be said because someone, somewhere, doesn't realize this common sense stuff and would expect everything to work perfectly, in spite of their coding mistakes. Used with good coding and testing practices, the procedures documented in this book can be used to great advantage in the right applications.

Copyright (C) 1989 Propellerhead Publishing

How To Use This Manual

Generally, the use of this manual (unless you like just thumbing through manuals) will be driven by applications being written. You, the programmer, will reach a point in your coding where you need to perform a specific function dealing with the internals of MPE. You can accomplish this one of two ways. The first is to get out your dogeared Tables Manual, forage through the tables and figure out what table contains the data you need, then figure out how to traverse the other tables to get there given the input data you have available in your program.

The second way is to use COMPASS. Given the above scenario, you would need to find a procedure that performs the function you want. The prime resource for this is the "one liner" chart on page 2-4. This chart presents each procedure in alphabetical order, a keyword classification of that procedure (directory, IO, etc) and a one or two line description of the procedure. It is from this one-line description that you should be able to at least get in the ballpark and find the procedure that will perform the function you want.

For example, assume you have an application and you need to determine whether a particular ldev is a terminal or not. You could do this by opening the Tables Manual, turning to Chapter 13, and finding the device type and subtype in the LPDT and LDT tables. Then you write some privileged code to read the entries for the ldev, and hardcode into your program the proper subtypes and types for terminals.

The other way is to look at the table on page 2-??. Scanning the one line descriptions, you see that the procedure CHEKLDEV has a description of "Procedure to determine if the specified device is a terminal or serial I/O device". Sounds like a winner. You then look up the procedure in the Table of Contents or Index, and find it is on page 2-41.

Looking further at the description, you see that the procedure has one parameter, the device number which is an integer passed by value. The procedure returns whether the ldev is a terminal by setting CCG, a serial IO device by setting CCE. CCL indicates the device number is invalid.

From this, you can then define the procedure in your SPL program:

```
PROCEDURE CHEKLDEV (DEVICE);
  VALUE DEVICE; INTEGER DEVICE;
  OPTION EXTERNAL;
```

At the proper point in the code, you then call the procedure:

```
CHECKLDEV(MYLDEVNO);
IF < THEN HANDLE'INVALID'LDEV
ELSE IF = THEN HANDLE'SERIAL'DEVICE
ELSE IF > THEN HANDLE'TERMINAL'DEVICE;
```

That's all there is to it.

Now assume that you have determined that the ldev is a terminal, but you want to find out what kind of terminal it is, assuming it is an HP terminal. In this case, we are not really dealing with an MPE table consideration, but with manipulating the terminal. How do we determine what type of terminal it is? Well, some things have been written about it, but where? Looking through past proceedings could be a pain. So you pull out your copy of COMPASS.

Looking again at the chart on page 2-4, you see a procedure called IDENTIFY'TERMIN. The one liner for this says "Identify the model number of the HP terminal currently being used by the calling process." Reading further, it shows that you have to open the terminal as a file first before calling this procedure. The SPECIAL CONSIDERATIONS section shows that some FCONTROL's have to be issued first as well. Opening the terminal makes sense, since a status check has to be made on the terminal to get it to return it's id. Using this section from COMPASS, you write the code to identify the terminal correctly with a minimum of muss and fuss.

Copyright (C) 1989 Propellerhead Publishing

For bigger jobs, such as manipulation of the directory, reading the Expansion in Section III might be in order. The directory portion of this section will go through almost all the directory procedures, discussing the idiosyncrasies of the directory routines. Others Expansions on IO and security are included as well.

Most of the procedures documented are relatively straightforward to call. If after using this manual you feel that a procedure or group of procedures deserve further discussion, please fill out the Comments and Feedback sheet at the end of this manual. We will be glad to respond in the best manner possible.

Copyright (C) 1989 Propellerhead Publishing

Section II - Procedures

Introduction and Conventions

This section defines the various low level procedures of MPE-V. The procedures are presented alphabetically, and all follow the same basic format:

Heading - This defines the procedure name.

Purpose - A one or two line description of what the procedure is designed to accomplish.

Syntax - A shaded box showing the procedure calling sequence for that procedure. The abbreviations used are outlined below.

Functional Return - (Optional) If the procedure is a typed function and returns a value, this section will contain a description of the data returned by the procedure.

Parameters - A description of each parameter, the possible values, and returned data if appropriated. This section is usually the lengthiest, and will contain most of the important data for each procedure.

Condition Codes - If the procedure sets the hardware condition code (CC) values, this will outline the meanings of CCE, CCL, and CCG. If the condition code is unaffected, this will be indicated.

Special Considerations - This will define any unusual or unique facets of that procedure.

Discussion - A few notes or side issues to be considered. If other procedures relate to the current one, these will be listed here.

Abbreviations

Each procedure is defined with a shaded box showing the procedure calling sequence. The one following is for the procedure DIRECLOGOFF:

```
I                              IV    LA  DV      DV LA
result = DIRECLOGOFF(mask, jmat, connect, cpu, account,
                               LA  LA
                               user, group)  O-P
```

DIRECLOGOFF is a good example, as it shows a functional return value as well as several different kinds of parameters.

The procedure name is in all caps (ie, DIRECLOGOFF). The functional return parameter is shown in lower case and normal typeface. The parameters are also shown in lower case, but are always in italics, such

Copyright (C) 1989 Propellerhead Publishing

as *mask* and *connect*. Above each parameter name is a one or two character abbreviation. The first character indicates the type of each parameter. The first name listed below is the SPL name, the name in parenthesis indicates the type for PASCAL:

Letter	Meaning
I	Integer (shortint.. user must define)
L	Logical (shortint.. user must define)
D	Double (integer)
B	Byte (PAC)
R	Real (Real)

The second letter, if present, indicates other data about the parameter:

Letter	Meaning
V	Parameter is passed by VALUE (no VAR)
A	Parameter is an Array
P	Parameter is a Procedure PLABEL

If the second letter is missing, then the parameter is passed by reference. That is, the address of the parameter is passed, and it is possible for the procedure to return values in that parameter.

To declare the procedure, use the lettered abbreviations. Some translation will be necessary for the specific language. The examples below show the DIRECLOGOFF procedure translated to both SPL and PASCAL:

```
SPL:
     INTEGER PROCEDURE DIRECLOGOFF(MASK,JMAT,CONNECT,
                                   CPU, ACCOUNT, USER,
                                   GROUP);
       VALUE MASK,  CONNECT, CPU;
       INTEGER MASK;
       DOUBLE CONNECT, CPU;
          LOGICAL ARRAY JMAT, ACCOUNT, USER, GROUP;
          OPTION EXTERNAL;

PASCAL:
     TYPE
          LOGARRTYPE      = ARRAY [1..20] OF SHORTINT;
          SHORTINT        = -32768..32767;
     .
     .
     FUNCTION DIRECLOGOFF(     MASK    : SHORTINT;
                      VAR  JMAT    : LOGARRTYPE;
                           CONNECT : INTEGER;
                        CPU      : INTEGER;
                     VAR   ACCOUNT : LOGARRTYPE;
                     VAR   USER    : LOGARRTYPE;
                     VAR   GROUP   : LOGARRTYPE):
                                          SHORTINT;
     EXTERNAL SPL;
```

Note that it will, in general, be difficult to call some procedures from PASCAL-V due to the strong typing features of PASCAL.

Copyright (C) 1989 Propellerhead Publishing

Different typefaces indicate different things as well. Headings are in a very large type **(Heading)**. Subheadings in a smaller typeface **(Subheading)**. Tables are generally in a non-proportional font (Non-Prop). Parameter names are in an *italic typeface*. Any sample code, like the SPL and PASCAL samples above, are in a courier typeface (Courier).

Procedure Options

After the listing of the procedure name, the parameters, and parameter types, a couple of different options might appear listed on the procedure. They are:

Option	Meaning
O-P	Option Privileged. This means the caller must be in Privileged Mode to call this procedure.
O-V	Option Variable. This means that parameters may be omitted from the call. The called procedure will generally check whether the parameters are present or not.

In SPL, procedures listed with O-V must be declared with the OPTION VARIABLE statement in the EXTERNAL declaration. In Pascal, EXTERNAL VARIABLE SPL must be used.

Copyright (C) 1989 Propellerhead Publishing

Procedure Function Chart

Proc Name	Classification	Description
ABORT	MISC	Aborts the calling process (or process tree) and all descendants and issues a message indicating the cause the abort.
ABORTIO	IO	Procedure to abort I/O on the specified device.
ABORTJOB	MISC	Procedure to abort any job or session.
ABORTPROCIO	IO	Procedure to abort I/O for the specified process.
ABORTTIMEREQ	MISC	Aborts a timer request (if still active) and returns the en to the free list.
ACCCHECK	Security	Aborts a timer request (if still active) and returns the en to the free list.
ADDJTENTRY	MISC	Add or update file equations in the Job Directory Ta (JDT) for the calling process.
ADOPT	Process	Attaches a process to PROGEN and removes it a descendent to the calling process.
ALLORIN	MISC	Procedure to obtain RINs.
ALTDSEGSIZE	MISC	This procedure is used to alter the size of a data segme
ALTEROUTPUTFILE	Spooling	Alter the output characteristics of output spoolfiles.
ATTACHIO	IO	Transfer between memory and external devices.
AWAKE	MISC	Used to awaken a process waiting on the specified ev
BUILDSTACKMARKER	MISC	Builds a stack marker at the designated Q+ address for specified plabel.
BURRYPROC	Process	Procedure to abort any process.
C'CLOCK	Date	Returns system time in form HH:MM:SS.
C'DATE	Date	Returns system date in form MM/DD/YY.
CACHE'LDEV	Caching	Enables caching for the specified disc drive.
CDT'SET'POST	Caching	Enable/disable block on write for cache I/O.
CDT'SET'RND	Caching	Controls cache fetch quantum for random I/O.
CDT'SET'SEQ	Caching	Controls cache fetch quantum for sequential I/O.
CENTER	MISC	Centers text within a byte array.

Copyright (C) 1989 Propellerhead Publishing

Proc Name	Classi-fication	Description
CHANGEDB	MISC	Changes the process's DB register to point to the an extended memory address or specific data segment.
CHECK'BOT	IO	Indicates is the specified tape drive is at the beginning of the tape (BOT).
CHECKALIVE	Process	Determines the specified PIN is alive.
CHECKALLOW	Security	Indicates if the user has been granted the specified command via ALLOW.
CHECKDISC	IO	To determine if a given logical device is configured and whether or not it is a disc.
CHECKLDEV	IO	Procedure to determine if the specified device is a serial I/O device.
CHEK	MISC	Performs parameter checking, user capability checking and determines if DB is at the stack or not. If any check fails, the caller is aborted.
CLEANUP	Process	Closes all files, data bases, etc. opened by the calling process.
CLEAR'PSIF	Process	Procedure to reset the pseudo-interrupt flags for the specified PCB.
CLEARWWS	Process	Clears the WakeUp-Waiting Switch in the event word of the current process' PCB.
CLEARBREAK	Process	Performs programmatic clearing of break points from within the calling process.
CONSALTFILE	Spooling	Performs programmic altering of output spoolfiles and input stream jobs (using the secondary entry point CONSALTJOB).
CONVERTDATE	Date	Convert a date on the format MM/DD/YY into the same format returned by the CALENDAR intrinsic.
CONVERTTIME	Date	Convert the time in the format HH:MM into the same format returned by the TIMER intrinsic.
CONVEXTLABELTODELTAP	MISC	Returns the DELTAP of the entry point identified by the STT and segment number contained in the external label.
CRYPT	Security	Encrypts a password.
DATEJUL	Date	Converts gregorian date to julian.
DEALLORIN	MISC	Releases a RIN and returns it to the list of available RINs.
DEBLANK	MISC	Locate the last non-blank character in a byte array.

Copyright (C) 1989 Propellerhead Publishing

Proc Name	Classi-fication	Description
DELAY	MISC	Causes the calling process to suspend for the specif amount of time.
DELETEDEVFILE	Spooling	Delete input/output spoolfiles.
DELINKSCHED	Process	Removes a entry from the scheduled job queue.
DEVSPEC	IO	Provides the actual device class name associated witl device class index.
DIRECADJUST	Directory	To adjust the group and account file space counts.
DIRECFIND	Directory	Obtain entries from the system directory.
DIRECFINDFILE	Directory	Obtain file entries from the system directory.
DIRECINSERT	Directory	To insert a group, account, user or volume entry into directory and adjust the appropriate internal pointers.
DIRECINSERTFILE	Directory	To insert a file entry into the directory and adjust the grc and account sector counts.
DIRECLOGOFF	Directory	Updates the system directory (counters) along with cpu a connect time for the logging-off user.
DIRECLOGON	Directory	Updates the system directory (counters) to reflect the r logon user.
DIRECPURGE	Directory	To purge an account, group, user, or volume entry fr the directory.
DIRECPURGEFILE	Directory	To purge a file entry from the directory and adjust the grc and account file space counts.
DIRECSCAN	Directory	Locate entries in the system directory.
DIREC[RE]SETFLAG	Directory	To reset/set the Bad File Label flag in a file entry in directory.
DIRMATCH	Directory	Determines if the actual account structure entry na matches the designated name.
DISCSIZE	IO	Return the maximum size (address) of a particular log disc device.
DISCTYPE	IO	Given an disc LDEV, returns the disc volume label al with a number indicating the disc drive model.
DISMOUNT	Directory	To terminate a logical connection (mount) of a Priv Volume set and remove any binding information that r have been established by the corresponding MOUNT.
DSMOVE	MISC	Transfers data between the caller's stack and the speci data segment.

Copyright (C) 1989 Propellerhead Publishing

Proc Name	Classification	Description
DSOPEN	DS/Comm	Opens a DSLINE.
DST'SIZE	MISC	Determines the current size of the specified data segment.
ERROREXIT	Security	Part of the mechanism used to pass error abort information between procedures.
ERRORON	Security	Part of the mechanism used to pass error abort information between procedures.
EXCHANGEDB	MISC	Changes the process's DB register to point to the base of the specified data segment.
FALTSEC	Security	Obtains/changes the security matrix and/or security/release state of a file.
FAMILY	Process	Returns all direct and indirect sons of the specified process.
FGETLOCKWORD	Security	Returns the lockword of an open file.
FINDDEVICES	IO	Identifies the logical devices number(s) for all devices within the requested class.
FLABIO	Directory	Read/write a file label.
FMAKETEMP	Directory	Converts an opened permanent file into an opened temporary file.
FORMSG	MISC	Routes a message from the program or the system message catalog to a device, or to an internal buffer.
FORMUSERID	MISC	Forms the user ID for the calling user.
FREEZESEG'	MISC	Freezes the specified DST in memory and prevents memory manager from relocating it in memory.
FREPLY	MISC	Writes a message on $STDLIST and returns a reply from $STDIN.
FROMCALENDAR	Date	Converts year, day of year into month and day of month.
FSCLOSE	Spooling	Close/delete files contained in the Input/Output spool. (This is a secondary entry point into the FCLOSE intrinsic).
FSECTORS	Directory	Determines how many sectors are used by a file.
FSOPEN	Spooling	Entry point into FOPEN to access file contained in the input/output spool queues.
GENMSG	MISC	Prints an error message on the requested device.
GENMSGU	MISC	Prints the system error message on the user's terminal.
GET'DSDEVICE	DS/Comm	Returns information on potential DS devices.

Copyright (C) 1989 Propellerhead Publishing

Proc Name	Classi-fication	Description
GET'FILECODE	Directory	Returns the file code assigned to the specified mnemonic.
GET'FILEMNEMONI	Directory	Returns the mnemonic associated with the specified code.
GET'SYS'PRIV	MISC	Determines if the specified code segment is privileged system.
GETCLASS	IO	Obtains information on devices in a particular class.
GETCLOCK	Measurement	Initiate the shared clock interface and provide a u procedure that is performed when the interrupt occur
GETDATASEG	Misc	Acquires a data segment and returns its DST number.
GETDEVINFO	IO	Obtains the LDT, LPDT and DCT table entries for a devi
GETJITDST	MISC	Returns the DST number of the calling user's Information Table (JIT).
GETJOBN	MISC	Returns the job number of the calling user.
GETMAXJS	MISC	Returns the maximum number of jobs and sessions specif during system configuration.
GETPCBXMEASINFO	Process	Retrieve information for a PIN from its PCB entry and PCBX fixed area in the stack.
GETSALT	Security	Calculates the password encryption seed based upon user, group and/or account names.
GETSECINFO	Security	Obtains security configuration information from the syst global security DST.
GETSIR	Process	Obtains a lock on the specified system internal resou (SIR) number prior to modifying the associated data. If SIR is currently locked by another process, the call process will be impeded until the SIR becomes availal This prevents concurrent modification of data.
GETSTATISTICS	Measurement	Returns information from the MPE Internal Measurem Facility.
GIVECLOCK	Measurement	Releases the shared clock interface.
HELPPROC	MISC	This procedure is the HELP subsystem driver.
IDENTIFY'TERMIN	IO	Identify the model number of the HP terminal curre being used by the calling process.
IMPEDE	Process	Impedes the process located at the specified address in PCB table.

Copyright (C) 1989 Propellerhead Publishing

Proc Name	Classi-fication	Description
IOCONTROL	IO	Procedure to perform I/O control operations on the specified device.
IOFREEZE	IO	If the specified segment memory is in memory, it is locked to prevent swaps to virtual.
IOFREEZE'	IO	If the specified segment memory is in memory, it is locked to prevent swaps to virtual.
IOSTAT	IO	Returns the transmission log/status for a given ioq entry.
IOUNFREEZE	IO	Unlocks the specified segment to allow it to be swapped to virtual storage.
IOUNFREEZE'	IO	Unlocks the specified DST to allow it to be swapped to virtual storage.
ISSYSCACHABLE	Caching	Determines if disc caching is installed.
JULDATE	Date	Converts a julian date to gregorian.
LDEVTOTYPE	IO	Returns device type for specified LDEV number.
LDEVTOVTAB	IO	Converts a disc file extent map from logical device number to volume table indexes.
LOCKJIR	MISC	Resource lock facility used to coordinate access to a process's Job Directory Table (JDT) table.
LOCKSEG	MISC	Locks the specified segment in memory to prevent swaps to virtual.
LOCKSEG'	MISC	Locks the specified DST in memory to prevent swaps to virtual.
LOGICALCST	MISC	Returns the number and type associated with the CST contained in the status word.
LUN	IO	Returns the logical device number of a disc given its volume table index.
MAKEFN	Date	Generates a filename using the current date and time.
MASTEROP	MISC	Indicates if the current logon device is the system console.
MAX'DST'SIZE	MISC	Returns the maximum size initially specified for a DST.
MOUNT	Directory	To establish a logical connection (mount) a Private Volume set and return an identifying handle for use by other procedures.
MPE	MISC	Indicates the version of MPE in use.
MUSTOPEN	Security	In normal user mode, MUSTOPEN functions the same as FOPEN. In privileged mode, if MUSTOPEN is called against

Copyright (C) 1989 Propellerhead Publishing

Proc Name	Classi-fication	Description
		a file which has a lockword, the open will succeed ; the user is not prompted for the password.
MYPIN	Process	Obtains the Process Identification Number (PIN) assig to the calling process.
OBTAIN	MISC	Provides a method of managing (locking) access t resource.
PCB'INDEX	Process	Returns the PCB relative index for the specified PIN. A indicates the validity of the PIN and if the process is al
PHYSICALCST	MISC	Returns the CST number assigned to the specified segm
PINJOBMAP	Process	Given a PIN, this procedure returns the corresponding number and an array containing the formatted job numl
PROCESS'COLD'LOAD'INFO	MISC	Provides access to the extension area of the Cold L Information Table.
PROCFILE	Process	Obtain the name of the program file name currently be executed by a particular process.
PRODUCEPARMS	Directory	Parses fully-qualified name and sets up the parameters the procedure DIRECSCAN.
QUEUEPROC	Process	Used to add or remove a process from the dispatch que
RELDATASEG	MISC	Releases a data segment.
RELEASE	MISC	Unlocks a resource previously locked via OBTAIN.
RELSIR	Process	Releases the system internal resource (SIR) that obtained by a call to GETSIR.
REMOVE'PASSWORDS	Security	Removes passwords/lockwords from character strings.
REQUESTSERVICE	Process	Determines if break was hit or if the process has an at pending.
RESETBREAKBITS	IO	Procedure to reset BREAK and CONTROL-Y bits in LPDT for the specified device.
RESETCRITICAL	Process	Reverses the condition established by the SETCRITIC procedure. Any pending aborts on the process can then t place.
RESETDB	MISC	Sets the DB register to a specific address in bank 0, retu to the split-stack DST or process's stack.
RLOCK	MISC	Obtain a lock on the specified RIN.
RUNLOCK	MISC	Unlocks the specified RIN.

Copyright (C) 1989 Propellerhead Publishing

Proc Name	Classi-fication	Description
SCHEDDAYTIME	Date	Formats the specified date and time into a 14 character string.
SCHEDULESCHED	MISC	Links a JMAT entry into the appropriate slot in the schedule job queue.
SECHK'PASSWORD	Security	Determines if a password is valid for the specified user and account names.
SECURITYINSTALLED	Security	Indicates if HP's security product is installed.
SEGMENTER	MISC	Provides programmatic access to the SEGMENTER procedure.
SET'BREAKPNT	MISC	Performs programmatic setting of break points from within the calling processes.
SET'PSIF	Process	Procedure to set the pseudo-interrupt flags for the specified PCB.
SETCRITICAL	Process	Prevents the current running process from being aborted during the execution of a critical series of code.
SETDBTOSTACK	Process	Sets the calling process's DB register to the base address of the process's stack DST.
SETSYSDB	Process	Sets the calling process's DB register to the system global DB (%1000).
SFINDODD	Spooling	Returns the XDD segment relative address for the specified XDD entry.
SHUTDOWNCACHING	Caching	Stops caching on all discs.
STACK'DST	Process	Obtains the DST number of the calling process' stack.
STACKCHECK	Process	Determines if specified DST is a process stack.
STARTSTATISTICS	Measurement	Initiates the MPE Internal Measurement Facility.
STOPSTATISTICS	Measurement	Terminates the MPE Internal Measurement Facility.
STRINGPINATTAIL	Process	Places the specified PIN at the end of the list of impeded processes strung together via PCB entries.
SYSGLOB	MISC	Returns a specified word from the System Global Area or the System Global Extension Area.
SYSPROC	Process	Converts a PIN number to a PCB index.
THISCPU	MISC	Obtain a code representing the type of computer on which the program is being executed.
TIMEREQ	MISC	Sets up a time to expire after a specified interval.

Copyright (C) 1989 Propellerhead Publishing

Proc Name	Classi-fication	Description
TOCALENDAR	Date	Given year, month and day, determined day of year.
UDC	MISC	Executes User Defined Commands (UDCs).
UNCACHE'LDEV	Caching	Disables caching for the specified disc drive.
UNFREEZESEG'	MISC	Unfreezes the specified DST from memory and all memory manager from relocating it in memory.
UNIMPEDE	Process	Unimpedes the process located at the specified address the PCB table.
UNLOCKJIR	Process	Releases the process's JIR.
UNLOCKSEG	MISC	Unlocks the specified segment to allow it to be swapp to virtual storage.
UNLOCKSEG'	MISC	Unlocks the specified DST to allow it to be swapped virtual storage.
UNSTRINGHEADPIN	Process	Removes the specified PCB entry from the string of P entries on a waiting (impeded) list.
VERIFY'PASSWORD	Security	Verifies the match of a password with that contained in directory entry. Since password encryption may be enabl the procedure may have to encrypt the password bef making the comparison.
VTABINX	IO	Return the volume table index assigned to the specified c device.
VTABTOLDEV	IO	Converts a disc file extent map from volume table inde to logical device numbers.
WAIT	Process	Used to block the calling process until the specified ev occurs.
WHERES'DB	Process	Determines where the DB register is currently pointin
XCODETRAP	MISC	Enables or disables code traps.
XRETJTENTRY	MISC	Returns a file equation entry from the JDT.

Copyright (C) 1989 Propellerhead Publishing

ABORT

PURPOSE

Aborts the calling process (or process tree) and all its descendants and issues a message indicating the cause of the abort.

SYNTAX

```
                    LV  LV  LV
ABORT(mode, code, parm)  O-P
```

PARAMETERS

mode (Logical by value, required)
This parameter indicates the number of stack markers to be deleted from the stack trace (bits 0:8) and the type of abort to be performed (bits 8:8) as follows:

Value	Type Of Error
0	Internal interrupt error. The code parameter indicates the type of error that occurred.
1	Intrinsic error. The code parameter contains the intrinsic information and the parm parameter contains the error type.
2	The process called the QUIT intrinsic. The parameter contains the user supplied quit number.
3	The process called the QUITPROC intrinsic. The parm parameter will contain the user supplied quit number.
4	Stack overflow.
5	Hard kill, such as :ABORT.

code (Logical by value, required)
If mode.(8:8) = 1, this parameter indicates the type of internal interrupt that occurred as follows:

Copyright (C) 1989 Propellerhead Publishing

Code	Error
0	(Unknown)
1	Integer overflow.
2	Floating point overflow.
3	Floating point underflow.
4	Integer divide by zero.
5	Floating point divide by zero.
6	Privileged instruction
7	(Unknown)
8	Extended precision overflow.
9	Extended precision underflow.
10	Extended precision divide by zero.
11	Decimal overflow.
12	Invalid ASCII digit in decimal instruction.
13	Invalid decimal digit.
14	Invalid source word count.
15	Invalid decimal operand length.
16	Decimal divide by zero.
17	STT uncallable.
18	(Unknown)
19	(Unknown)
20	Stack overflow.
21	(Unknown)
22	Bad stack marker.
23	Illegal address.
24	Bounds violation
25	(Unknown)
26	(Unknown)
27	(Unknown)
28	(Unknown)
29	Stack overflow.
30	CST violation.
31	STT violation.

If mode.(8:8) = 1, this parameter provides information about the intrinsic being called:

Bits	Contents
0:10	Intrinsic number.
10:6	Number of parameter words.

For all other modes, this parameter is not used.

parm (Logical by value, required)
For mode.(8:8) = 1, this parameter contains the type of intrinsic error that occurred:

Code	Type Of Error
1	Illegal DB register.
2	Illegal capability.

Copyright (C) 1989 Propellerhead Publishing

Code	Type Of Error
3	Ommitted parameters
4	Not enough stack.
5	Parameter address error
6	Parameter end address error.
7	Illegal parameter.
8	Invalid parameter value.
9	Illegal Q register.

For mode.(8:8) = 2 or 3, this parameter contains the user-supplied quit number. This parameter is ignored for all other modes.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

ABORTIO

PURPOSE

Procedure to abort I/O on the specified device.

SYNTAX

```
                IV
ABORTIO(device) O-P
```

PARAMETERS

device (Integer by value, required)
The number of the logical device to abort all pending I/O. If a negative device number is supplied, only I/O related to the calling PIN will be aborted.

CONDITION CODES

CCL	Invalid LDEV.
CCE	Successful.
CCG	No I/O entries found

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

None.

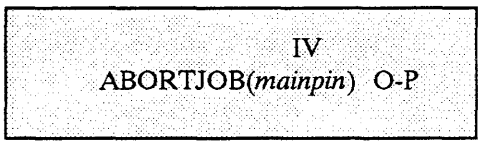
Copyright (C) 1989 Propellerhead Publishing

ABORTJOB

PURPOSE

Procedure to abort any job or session.

SYNTAX



PARAMETERS

mainpin (Integer by value, required)
The main PIN of the job or session that is to be aborted.

CONDITION CODES

CCL PIN is not a main pin.
CCE Successful.
CCG Main PIN is not alive

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

If this procedure is used to abort the calling user, perform a SETCRITICAL before calling ABORTJOB. This will prevent a system hang caused by the procedure filling the IOQ table. After returning from ABORTJOB call RESETCRITICAL at which point the user will be aborted.

Copyright (C) 1989 Propellerhead Publishing

ABORTPROCIO

PURPOSE

Procedure to abort I/O for the specified process.

SYNTAX

PARAMETERS

pin (Integer by value, required)
The PIN number of the process to abort all pending I/O.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

None.

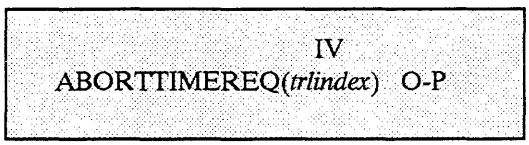
Copyright (C) 1989 Propellerhead Publishing

ABORTTIMEREQ

PURPOSE

Aborts a timer request (if still active) and returns the entry to the free list.

SYNTAX



PARAMETERS

trlindex (Integer by value, required)
Entry number into the TRL returned by the TIMEREQ procedure.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

Potential system failures include:

```
24 - Index not within the table.
28 - Entry already free.
```

Copyright (C) 1989 Propellerhead Publishing

ACCCHECK

PURPOSE

To determine what access a user is allowed to a file, group or account.

SYNTAX

```
 L                          IV   BA   LV   BA   DV
access = ACCCHECK(level, acct, acctsec, group, grpsec,
                            BA   DV   BA
                  creator, filesec, userinfo) O-V, O-P
```

FUNCTIONAL RETURN

The returned logical value indicates the level of access permitted as follows:

Bits	Access
10:1	Read
11:1	Append
12:1	Write
13:1	Lock
14:1	Execute
15:1	Save

PARAMETERS

level (Integer by value, required)
The level of access being examined as follows:

Code	Level
0	File
1	Group
2	Account

acct (Byte array, required)
The name of the account being examined.

acctsec (Logical by value, required)
The security access assigned to the specified account.

group (Byte array, optional)
A byte array containing the name of the group to be examined. Required parameter if level is <= 1.

Copyright (C) 1989 Propellerhead Publishing

grpsec (Double by value, optional)
The security attributes assigned to the specified group. Required parameter if level is <= 1.

creator (Byte array, optional)
A byte array containing the name of the creator of the file. Required parameter if level is equal to 0.

filesec (Double by value, optional)
The security attributes assigned to the specified file. Required parameter if level is equal to 0.

userinfo (Byte array, optional)
A byte array containing the account, home group, logon group and name of the user whose access is being examined. If omitted, this information is obtained from the process's Job Information Table (JIT).

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

ADDJTENTRY

PURPOSE

Add or update file equations in the Job Directory Table (JDT) for the calling process.

SYNTAX

```
 I                          BABA  BA  IV    IV  IA
flag = ADDJTENTRY(file, grp, acct, table, size, eq)  O-P
```

FUNCTIONAL RETURN

An flag indicating successful (0), out of space (1) or duplicate entry (2). A value of 2 is not returned when updating the table.

PARAMETERS

file (Byte array, required)
The formal file name.

grp (Byte array, required)
The formal group name.

acct (Byte array, required)
The formal account name.

table (Integer by value, required)
This variable indicates which table is to be accessed:

```
1 - Data Segment Directory
2 - Temporary File Directory
3 - File Equation Table
```

size (Integer, required)
The size of the equation array to be stores in the table.

eq (Integer array, required)
An array that contains the file equation to be added or updated. See the System Table Manual for the format of the file equation entry with the PMASK.

CONDITION CODES

Unchanged.

Copyright (C) 1989 Propellerhead Publishing

ADDJTENTRY (continued)

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

ADOPT

PURPOSE

Attaches a process to PROGEN and removes it as a descendent to the calling process.

SYNTAX

```
                I              LV      LV
error = ADOPT(pin, new'parent)  O-P
```

FUNCTIONAL RETURN

The integer value returned from the procedure is as follows:

Value	Meaning
0	Successful.
1	Invalid PIN specified for new parent
2	Can't adopt a process when current father is a system process.

PARAMETERS

pin (Logical by value, required)
The PIN number of the son process to be attached to a different father process.

new'parent (Logical by value, required)
The pin number of a current process which is to become the new father of the specified pin. Currently the only accepted value is the PIN number of PROGEN (PIN 1).

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode. Once the process has been adopted to the new parent, it is activated using AWAKE and specifying a father wait event.

DISCUSSION

There exists an additional entry point to ADOPT called ONENET'ADOPT. This entry point allows a new parent other than PROGEN.

Copyright (C) 1989 Propellerhead Publishing

ALLORIN

PURPOSE

Procedure to obtain RINs.

SYNTAX

```
 I                  IV    LA      LA
rin = ALLORIN(code, username, password)  O-P, O-V
```

FUNCTIONAL RETURN

The ALLORIN procedure returns the RIN number. A zero RIN number returned indicates no RINs are available.

PARAMETERS

code (Integer by value, required)
A code indicating the type of RIN to be obtained and if the calling process's DB register to already pointing to the RIN table. This parameter is constructed as follows:

Bits	Value
0:1	If set, then DB is pointing to the RIN table. Can be done when obtaining local and file RINS only.
1:15	A flag defining the RIN type: 1 - Local RIN 2 - Global RIN 3 - File RIN

username (Logical array, optional)
A logical array containing the name of the user that will own the global RIN. This parameter is required when obtaining a global RIN. The array is constructed as follows:

Words	Contents
0-3	The user name, left-justified with trailing spaces
4-7	The user's account name, left-justified with trailing spaces

password (Logical array, optional)
A logical array containing the password assigned to the global RIN. The password must be left-justified in the array and contain trailing spaces.

Copyright (C) 1989 Propellerhead Publishing

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

The user is not required to have MR capability to obtain multiple RINs. It is possible to obtain a global RIN without assigning any password or user to the RIN. Compliments the procedure DEALLORIN.

Copyright (C) 1989 Propellerhead Publishing

ALTDSEGSIZE

PURPOSE

This procedure is used to alter the size of a data segment.

SYNTAX

```
                I                          IV      IV
newsize = ALTDSEGSIZE(dst, adjustment)  O-P
```

FUNCTIONAL RETURN

The ALTDSEGSIZE procedure returns an integer value equal to the new size of the modified data segment.

PARAMETERS

dst (Integer by value, required)
The DST number of the data segment to be modified.

adjustment (Integer by value, required)
The number of words to increase (positive) or decrease (negative) the data segment. This value will be a rounded upwards to a multiple of 4 words and must be within the limit of virtual memory allocated to the segment when it was created.

CONDITION CODES

CCL Invalid DST or DST may be locked or frozen in memory.
CCE Successful.
CCG Adjustment would cause an invalid DST size or the DST to exceed its originally allocated virtual memory.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

Potential system failures include:

```
634 - PDISABLE count > 0.
```

Copyright (C) 1989 Propellerhead Publishing

ALTEROUTPUTFILE

PURPOSE

Alter the output characteristics of output spoolfiles.

SYNTAX

```
  I                                  IV IV IV    IV
error = ALTEROUTPUTFILE(dfid, pri, dev, copies) O-P O-V
```

FUNCTIONAL RETURN

The integer value returned from the procedure is as follows:

Value	Meaning
0	Successful.
-1	Output spoolfile does not exists.
-2	Invalid or missing output spoolfile ID.
>0	Output spoolfile currently active.

PARAMETERS

dfid (Integer by value, required)
The number assigned to the output spoolfile.

pri (Integer by value, optional)
A new priority for the output spoolfile.

dev (Integer by value, optional)
New output device number for the output spoolfile. This should be the same format as returned from the GETDEVINFO procedure.

copies (Integer by value, optional)
Number of copies of the specified output spoolfile.

CONDITION CODES

CCE	Successful.
CCL	The output spoolfile ID is either invalid, not specified, or doesn't exist.
CCG	The specified output spoolfile is active.

Copyright (C) 1989 Propellerhead Publishing

ALTEROUTPUFILE (continued)

SPECIAL CONSIDERAS

Requires privileged mode.

DISCUSSION

Note, there is a special format pvice parameter. Refer to the GETDEVINFO procedure.

C) 1989 Propellerhead Publishing

ATTACHIO

PURPOSE

Transfer between memory and external devices.

SYNTAX

```
                D                              IV   IV    IV   IV   IV
d'status := ATTACHIO (ldev, qmisc, dstx, addr, funct,
                      IV   IV IV   IV
                      count, p1, p2, flags)  O-P
```

FUNCTIONAL RETURN

For blocked I/O request:

Word	Bits	Description
0	0:7	PIN number.
	8:5	Qualifying status.
	13:3	General status: 0 - Pending 1 - Successful 2 - End of file 3 - Unusual condition 4 - Irrecoverable error
1		Number of words (positive) or bytes (negative) transferred.

For unblocked I/O request:

Word	Bits	Description
0		I/O queue index used by the procedure IOSTATUS to determine to determine if the request has completed.
1		Always zero.

Copyright (C) 1989 Propellerhead Publishing

PARAMETERS

ldev (Integer by value, required)
Logical device Number.

qmisc (Integer by value, required)
Miscellaneous parameter which may be required by the device. If not required, must be zero.

dstx (Integer by value, required)
This is the DST number of the data segment. If zero, then the 'addr' parameter contains a DB-relative address in the calling process's stack.

addr (Integer by value, required)
This is an offset into the data segment, user's stack or system buffer where the data to be transferred is located or to be placed.

funct (Integer by value, required)
Device dependent code indicating the type of operation to be performed. The usual codes are:

Code	Action	Code	Action
0	Read	%20	Disable Timer
1	Write	%21	Enable Timer
2	File open	%22	Read Timer
3	File close	%23	Disable parity
4	Device close	%24	Enable parity
5	Set Timeout	%25	Logged On
6	Set Inspeed	%26	Set parity
7	Set Outspeed	%27	Set Terminal Type
%10	Echo On	%30	Allocate Terminal
%11	Echo Off	%31	Clear Flush & Write
%12	Disable Break	%32	Enable Ctl-X !!! Echo
%13	Enable Break	%33	Disable Ctl-X !!! Echo
%14	Disable Escape	%34	(Not used)
%15	Enable Escape	%35	PTAPE Read
%16	Disable Tapemode	%36	Set/Reset Break Mode
%17	Enable Tapemode	%37	Set/Reset Console Mode

count (Integer by value, required)
Amount of data to transfer. A positive value indicates the number of words, a negative value indicates the number of bytes.

p1 (Integer by value, required)
Device dependent parameter.

p2 (Integer by value, required)
Device dependent parameter.

flags (Integer by value, required)
Control flags defined as follows:

Copyright (C) 1989 Propellerhead Publishing

Bits	Description
0:4	Callers ID: 0 - Unknown. 1 - File system. 2 - SPOOLER

Bits	Description
4:3	Unused.
7:2	0 - No preemption 1 - Soft preemption 2 - Hard preemption
9:1	Zero
10:1	If set, device dependent special request.
11:1	If set, diagnostic request.
12:1	If set, 'addr' is an index into the SBUF table.
13:3	Type of transfer request: 0 - Unblocked, no wake on completion, impede if no available I/O queue entries. 1 - Blocked, wait until request is complete (normal value) 2 - Unblocked, wake on completion, impede if no available I/O queue entries. 3 - Unblocked, no process associated with request, impede if no available I/O queue entry. 4 - Unblocked, no wake on completion, do not impede if no I/O queue entry available. 5 - Reserved. 6 - Unblocked, wake on completion, do not impede if no I/O queue entry available. 7 - Unblocked, no process associated with request, do not impede if no I/O queue entry available.

CONDITION CODES

Unchanged.

Copyright (C) 1989 Propellerhead Publishing

ATTACHIO (continue)

SPECIAL CONSIDERATIONS

Requires privileged mode, traps must be disabled before calling.

DISCUSSION

It is necessary to obtain the device-dependent information from the I/O driver in order to properly configure the ATTACHIO call. Refer to the Expansions Section (Section III) for more detail on ATTACHIO and its functional returns and parameters.

Copyright (C) 1989 Propellerhead Publishing

AWAKE

PURPOSE

Used to awaken a process waiting on the specified event.

SYNTAX

```
                    IV      LV      IV
AWAKE(pcbptr, eventmask, wait)  O-P
```

PARAMETERS

pcbptr (Integer by value, required)
This contains the PCB base address of the process to be awakened.

eventmask (Logical by value, required)
This is the event that the process is waiting on before it can continue executing. This bits in this mask are in the same order as those in the wakeup mask in the PCB table but are shifted right 4 bits. The following is a layout:

Bit	Event
0	Impeded
1	SIR Wait
2	Timeout
3	Memory Wait
4	Mourning Wait
5	Global RIN Wait
6	Local Rin Wait
7	Mail Wait
8	Blocked I/O Wait
9	I/O Wait
10	UCOP, RIT Wait
11	Junk Wait
12	Timer Wait
13	IPC Message Wait
14	Son Wait
15	Father Wait

Wait (Integer by value, required)
The wait condition flags that the caller wishes to wait for in case the PIN to be scheduled is not active (see WAIT).

Copyright (C) 1989 Propellerhead Publishing

AWAKE (Continue)

CONDITION CODES

CCL	The process is waiting, but not on the specified event.
CCE	The process was waiting on the specified event.
CCG	The process was already active.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

Potential system failures include:

```
610 - Being awaken for a critical event (SIR wait,
      device timeout or  memory) which the process
      wasn't expecting.
623 - Invalid PCB entry.
```

Copyright (C) 1989 Propellerhead Publishing

BUILDSTACKMARKER

PURPOSE

Builds a stack marker at the designated Q+ address for the specified plabel.

SYNTAX

```
                                    IV    LV
BUILDSTACKMARKER(loc, extlabel)  O-P
```

PARAMETERS

loc (Integer by value, required)
The Q positive relative address to construct the stack marker.

extlabel (Logical by value, required)
The external plabel of the procedure to be used in the stack marker.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

The Q relative address MUST be a positive address within the calling procedure's allocated stack area.

Copyright (C) 1989 Propellerhead Publishing

BURRYPROC

PURPOSE

Procedure to abort any process.

SYNTAX

```
                    IV
BURRYPROC(pinaddr) O-P
```

PARAMETERS

pinaddr (Integer by value, required)
The PCB address of the process to be aborted.

CONDITION CODES

Unchnaged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

All pending I/O for the specified process should be aborted by the ABORTPROCIO procedure prior to aborting the process. Soft-kill interrupt flag should be set on process also.

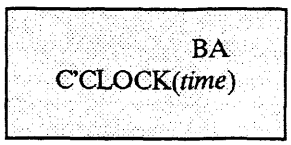
Copyright (C) 1989 Propellerhead Publishing

C'CLOCK

PURPOSE

Returns system time in form HH:MM:SS.

SYNTAX



PARAMETERS

time (Byte array, required)
The current system time in the form HH:MM:SS.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

None.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

C'DATE

PURPOSE

Returns system date in form MM/DD/YY.

SYNTAX

PARAMETERS

date (Byte array, required)
The current system date in the form MM/DD/YY.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

None.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

CACHE'LDEV

PURPOSE

Enables caching for the specified disc drive.

SYNTAX

```
                          IV   IV
CACHE'LDEV(ldev, status)  O-P
```

PARAMETERS

ldev (Integer by value, required)
The logical device number of the disc drive to be cached.

status (Integer by value, required)
A returned value indicating the status of the call. The possible values are:

Value	Meaning
0	Successful.
1	Could not obtain cache DST.
2	Internal cache error occurred.
3	The disc drive specified is already cached.
4	Unable to expand cache directory table.
5	Device is not cachable.
6	Caching is not installed.

CONDITION CODES

Not returned.

SPECIAL CONSIDERATIONS

Requires privileged mode. Do not call in split-stack mode.

DISCUSSION

See procedure UNCACHE'LDEV.

Copyright (C) 1989 Propellerhead Publishing

CDT'SET'POST

PURPOSE

Enable/disable block on write for cache I/O.

SYNTAX

```
                                IV    IV
CDT'SET'POST(parm, status)      O-P
```

PARAMETERS

parm (Integer by value, required)
A non-zero value indicates that the process should not be allowed to continue until the physical write has completed. A zero value allows the process to continue even it the physical write has not been completed.

status (Integer by value, required)
A returned value indicating the status of the call. The possible values are:

Value	Meaning
0	Successful.
1	Caching not enabled.
2	(Not used).
3	(Not used).
4	(Not used).
5	(Not used).
6	Caching is not installed.

CONDITION CODES

Not returned.

SPECIAL CONSIDERATIONS

Requires privileged mode. Do not call in split-stack mode.

DISCUSSION

See procedures CDT'SET'SEQ and CDT'SET'RND.

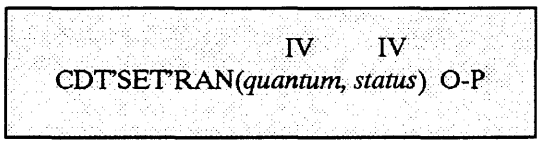
Copyright (C) 1989 Propellerhead Publishing

CDT'SET'RND

PURPOSE

Controls cache fetch quantum for random I/O.

SYNTAX



PARAMETERS

quantum (Integer by value, required)
A number of sectors to be transfered for random I/O. Acceptable values are from 1 to 96.

status (Integer by value, required)
A returned value indicating the status of the call. The possible values are:

Value	Meaning
0	Successful.
1	Caching not enabled.
2	(Not used).
3	(Not used).
4	(Not used).
5	(Not used).
6	Caching is not installed.

CONDITION CODES

Not returned.

SPECIAL CONSIDERATIONS

Requires privileged mode. Do not call in split-stack mode.

DISCUSSION

See procedures CDT'SET'SEQ and CDT'SET'POST.

Copyright (C) 1989 Propellerhead Publishing

CDT'SET'SEQ

PURPOSE

Controls cache fetch quantum for sequential I/O.

SYNTAX

```
                                IV      IV
CDT'SET'SEQ(quantum, status)  O-P
```

PARAMETERS

quantum (Integer by value, required)
A number of sectors to be transfered for sequential I/O. Acceptable values are from 1 to 96.

status (Integer by value, required)
A returned value indicating the status of the call. The possible values are:

Value	Meaning
0	Successful.
1	Caching not enabled.
2	(Not used).
3	(Not used).
4	(Not used).
5	(Not used).
6	Caching is not installed.

CONDITION CODES

Not returned.

SPECIAL CONSIDERATIONS

Requires privileged mode. Do not call in split-stack mode.

DISCUSSION

See procedures CDT'SET'RND and CDT'SET'POST.

Copyright (C) 1989 Propellerhead Publishing

CENTER

PURPOSE

Centers text within a byte array.

SYNTAX

```
I                    BA IV
size = CENTER(buffer, len)
```

FUNCTIONAL RETURN

The number of bytes (characters) that were centered.

PARAMETERS

buffer (Byte array, required)
The byte array containing the text to be centered.

len (Integer by value, required)
The length of the buffer.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

None.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

CHANGEDB

PURPOSE

Changes the process's DB register to point to the an extended memory address or specific data segment.

SYNTAX

```
 D                                DV
cur'addr = CHANGEDB(address)  O-P
```

FUNCTIONAL RETURN

The CHANGEDB procedure returns a double value equal to address or DST of the current DB register.

PARAMETERS

address (Double by value, required)
The address to set in the DB register. This double-word value is constructed as follows:

Value	Meaning
0,-dst	Address is a DST number.
0	Reset DB register to back to the process stack.
0,1	Reset DB to previous DST or stack.
>1	Bank and base address to set DB.

Word	Contents
0	Bank Number
1	Base Address

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

Copyright (C) 1989 Propellerhead Publishing

DISCUSSION

This procedure provides the most general method for moving DB. EXCHANGEDB only deals with data segment numbers, RESETDB deals only with absolute addresses in bank 0 and SETSYSDB deals only with address %1000 (SysGlob) in bank 0.

Copyright (C) 1989 Propellerhead Publishing

CHECK'BOT

PURPOSE

Indicates is the specified tape drive is at the beginning of the tape (BOT).

SYNTAX

```
 L                    IV
bot = CHECK'BOT(ldev) O-P
```

FUNCTIONAL RETURN

Return value TRUE if tape drive at beginning of tape, FALSE otherwise.

PARAMETERS

ldev (Integer by value, required)
The device number of the tape drive to be examined.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

CHECKALIVE

PURPOSE

Determines the specified PIN is alive.

SYNTAX

```
   L                          IV
alive = CHECKALIVE(pin)  O-P
```

FUNCTIONAL RETURN

A value of TRUE if the specified PIN is alive, FALSE is not.

PARAMETERS

pin (Integer by value, required)
The PIN number of the process to be examined.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

CHECKALLOW

PURPOSE

Indicates if the user has been granted the specified command via ALLOW.

SYNTAX

```
L                                IV
allow = CHECKALLOW(command)  O-P
```

FUNCTIONAL RETURN

Return value TRUE if specified command has been allowed, FALSE otherwise.

PARAMETERS

command (Integer by value, required)
A number identifying the command to be tested.as follows:

Value	MPE Command	Value	MPE Command
0	ABORTIO	24	OUTFENCE
1	ACCEPT	25	RECALL
2	DOWN	26	RESUMEJOB
3	GIVE	27	RESUMESPOOL
4	HEADOFF	28	STREAMS
5	HEADON	29	CONSOLE
6	REFUSE	30	WARN
7	REPLY	31	WELCOME
8	STARTSPOOL	32	MON
9	TAKE	33	MOFF
10	UP	34	VMOUNT
11	MPLINE	35	LMOUNT
12	DSCONTROL	36	LDISMOUNT
13	ABORTJOB	37	MRJECONTROL
14	ALLOW	38	JOBSECURITY
15	ALTSPOOLFILE	39	DOWNLOAD
16	ALTJOB	40	MIOENABLE
17	BREAKJOB	41	MIODISABLE
18	DELETESPOOLFILE	42	LOG
19	DISALLOW	43	FOREIGN
20	JOBFENCE	44	IMFCONTROL
21	LIMIT	45	SHOWCOM
22	STOPSPOOL	46	OPENQ
23	SUSPENDSPOOL	47	SHUTQ
		48	DISCRPS

Copyright (C) 1989 Propellerhead Publishing

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

None.

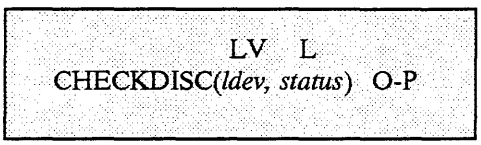
Copyright (C) 1989 Propellerhead Publishing

CHECKDISC

PURPOSE

To determine if a given logical device is configured and whether or not it is a disc.

SYNTAX



PARAMETERS

ldev (Logical by value, required)

The logical device number of the device being tested.

status (Logical, required)
The status of the device returned. The following bits on represent the indicated conditions:

Bits	Description
13:3	= 0: Device is disc. = 1: Invalid LDEV. = 2: Virtual LDEV. = 3: LDEV not disc.
12:1	Device does not support removable discs.
11:1	Device is not usable as a private volume.
10:1	Device in the system domain.
9:1	Device is offline.
8:1	Device is a serial disc and is not downed.
7:1	Device is reserved.
6:1	Device is downed.
5:1	Device has a 'DOWN' pending.
4:1	Device has a disc mounted.
3:1	Device is a foreign disc and is not downed.

CONDITION CODES

Unchanged.

Copyright (C) 1989 Propellerhead Publishing

CHECKDISC(continued)

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

Prior to T-MIT, calling CHECKDISC did not validate LDEV numbers.

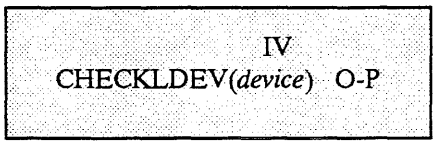
Copyright (C) 1989 Propellerhead Publishing

CHECKLDEV

PURPOSE

Procedure to determine if the specified device is a serial I/O device.

SYNTAX



PARAMETERS

device (Integer by value, required)
The number of the logical device to be checked.

CONDITION CODES

CCL Invalid device number.
CCE The specified device is a serial I/O device, disc, serial disc, cartridge tape or ramdisc.
CCG The specified device a non-disc deveice.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

In some instances the device is actually interrogated.

Copyright (C) 1989 Propellerhead Publishing

CHEK

PURPOSE

Performs parameter checking, user capability checking and determines if DB is at the stack or not. If any check fails, the caller is aborted.

SYNTAX

```
                D                    LV   LV   DV   DV
bounds = CHEK(intrinsic, flags, parms, capmask,
                                     LV
                              optvmask)  O-V, O-P
```

FUNCTIONAL RETURN

A double word value containing the lower and upper limits of the data segment. If the DB is at a stack, the limits will be DL -> S. If DB is pointing to an extra data segment, the limits will be from 0 -> end of the data segment.

PARAMETERS

intrinsic (Integer by value, required)
The intrinsic number and the number of parameters including the variable mask if present as follows:

Bits	Meaning
0:10	Intrinsic Number
10:6	Number of parameters, including the option variable mask.

flags (Logical by value, required)
This is the operational flags as follows:

Bits		Operation
0:1	= 0	DB must be pointing to stack.
	= 1	DB can be pointing anywhere.
8:2		Number of words for the intrinsic's functional return.
10:1	= 0	All capabilities specified in CAPMASK must be on.
	= 1	Any of the capabilities specified in CAPMASK must be on.

Copyright (C) 1989 Propellerhead Publishing

Bits	Operation
11:5	Number of formal parameters of the intrinsic to be checked.

parms (Double by value, required)
Indicates the bounds checking which should be performed. This parameter contains 16 2-bit fields, each describing the bounds checking of the parameter. The right-most field (bits 14-15 of the second word) describes the first parameter, the next right-most 2-bits describe the second parameter, etc.

The following values for the 2-bit fields are used to indicate the bounds checking to be performed:

Value	Type Of Checking
0	No checking for this single word parameter.
1	No checking for this double word parameter.
2	Check the bounds of this word address.
3	Check the bounds of this byte address.

If no bounds checking is to be performed this parameter should be omitted.

capmask (Double by value, optional)
Describes the capabilities needed by the intrinsic caller. Two types of capability checking can be performed according to bit 10 in the *flags* parameter. The high order word of this parameter contains the attribute flags to be tested and the low order word contains the resource flags. The following are the capability flags:

Bit	High-Order Word
0	SM - System Manager
1	AM - Account Manager
2	AL - Account Librarian
3	GL - Group Librarian
4	DI - Diagnostic Capability
5	OP - Operator Capability
6	CV - Create Volumes
7	UV - Use Volumes
8	LG - Logging Attribute
11	ND - Node Administrator
12	NM - Network Manager
13	CS - Access Communication Software
14	ND - Non-Sharable Device Access
15	SF - Save File Capability

Copyright (C) 1989 Propellerhead Publishing

Bit	Low-Order Word
7	BA - Batch Access
8	IA - Interactive Access
9	PM - Privileged Mode
12	MR - Multiple RINs
14	Extra Data Segments
15	PH - Process Handling

If no capability checking is to be performed, this parameter should be omitted.

optvmask (Logical by value, optional)
A mask indicating which parameters are optional. A bit is on for each parameter which is optional, with the least significant bit (bit 15) representing the last parameter, the next to the last parameter represented by bit 14, etc. If all parameters are required, this parameter should be omitted.

CONDITION CODES

CCL CHEK'NO'ABORT entry point was used and a check failed.
CCE Caller is not system code and all checks passed.
CCG Caller is system code and DB-at-stack check passed.

CARRY BIT

No Carry DB is pointing to the stack.
Carry DB is not pointing to the stack.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

A secondary entry point of CHEK'NO'ABORT exists which will not abort the caller upon detecting an error.

Copyright (C) 1989 Propellerhead Publishing

CLEANUPFILES

PURPOSE

Closes all files, data bases, etc. opened by the calling process.

SYNTAX

```
CLEANUPFILES O-P
```

PARAMETERS

None.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

CLEAR'PSIF

PURPOSE

Procedure to reset the pseudo-interrupt flags for the specified PCB.

SYNTAX

```
                                IV      LV
CLEAR'PSIF(pcbaddr, flag)       O-P
```

PARAMETERS

pcbaddr (Integer by value, required)
The PCB address of the process to be modified.

flag (Logical by value, required)
A value indicating which pseudo-interrupt is to be reset. The following are the valid pseudo-interrupt flags:

Flag	Pseudo-Interrupt
%40	Hard kill - process killed by UCOP or QUITPROG.
%20	Soft kill - process simply calls terminate.
%2	A Control-Y interrupt.
%1	A BREAK interrupt.

Any other value will cause a system failure.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

Copyright (C) 1989 Propellerhead Publishing

CLEAR'PSIF (continued)

DISCUSSION

Compliments the SET'PSIF procedure. The PCB must be valid and alive or a system failure will occur. DISPATCHER should be disabled during this call.

Copyright (C) 1989 Propellerhead Publishing

CLEARWWS

PURPOSE

Clears the WakeUp-Waiting Switch in the event word of the current process' PCB.

SYNTAX

```
CLEARWWS O-P
```

PARAMETERS

None.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

This procedure allows the waiting process an opportunity to check for events which happen prior to calling WAIT. Clearing this flag will result in any pending events to be ignored when the current process eventually goes into a wait state. This includes the critical event flags, SIRs, device timeout and memory events.

Copyright (C) 1989 Propellerhead Publishing

CLEAR'BREAKPNT

PURPOSE

Performs programmatic clearing of break points from within the calling process.

SYNTAX

```
                                    I    IV   IV    IV
CLEAR'BREAKPNT(error, seg, p'loc, segtype,
                                         LV    IV
                              break'type, label)  O-V
```

PARAMETERS

error (Integer, required)
An error code return if call was unsuccessful. Possible values are:

Value	Description
0	Successful.
1	Invalid program location.
2	Unused.
3	Unused.
4	Unused.
5	Unused.
6	Invalid segment.
7	Invalid segment type.
8	Invalid count.
9	Insufficient capability.
10	Invalid procedure label.
11	Invalid break point type.

seg (Integer by value, optional);
The segment in which the break point is to be cleared as follows:

Value	Description
-1	Clear ALL break points.
0	Specified break point is in the current segment.
>0	Segment which contains the break point to be cleared.

Copyright (C) 1989 Propellerhead Publishing

p'loc (Integer by value, required)
The PB relative location of the break point to be cleared as follows:

Value	Description
-1	Clear all break points in the specified segment.
>0	The PB relative location of the break point to be cleared.

Note, this parameter is not required if *seg* is equal to -1.

seg'type (Integer by value, optional)
The type of segment number contained in the *seg* parameter:

Value	Segment Type
0	Absolute memory segment.
1	System SL.
2	PUB SL.
3	Group SL.
4	Program segment.

If not specified, program segment is assumed.

break'type (Logical by value, optional)
This word describes the type of break point as follows:

Bits	Description
13:3	= 0 Local break point.
	= 1 System break point.
0:12	Ignored.

label (Integer by value, optional)
Which trap procedure are to be cleared as follows:

Value	Description
0	Remove ALL traps.
1	Remove only USER traps.
2	Remove only DEBUG traps.
Label	Remove trap routine that is identified by this label.

CONDITION CODES

CCL Failure.
CCE Successful.
CCG Not returned.

SPECIAL CONSIDERATIONS

None.

Copyright (C) 1989 Propellerhead Publishing

DISCUSSION

Compliments the procedure SET'BREAKPNT.

Copyright (C) 1989 Propellerhead Publishing

CONSALTFILE

PURPOSE

Performs programmic altering of output spoolfiles and input stream jobs (using the secondary entry point CONSALTJOB).

SYNTAX

FUNCTIONAL RETURN

The value returned indicates successful (TRUE) or failure (FALSE).

PARAMETERS

string (Byte Array, required)
This character string contains the normal console/MPE command used to alter either an output spoolfile or stream job. However, the command keyword itself is not included. For example:

```
#J10;INPRI=10
     or
#O1012;PRI=10;OUTDEV=6
```

All characters MUST be in upper-case.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

A secondary entry point, CONSALTJOB is used when the parameter string is for altering a stream job.

Copyright (C) 1989 Propellerhead Publishing

CONVERTDATE

PURPOSE

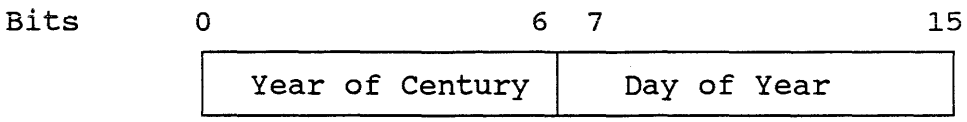
Convert a date on the format MM/DD/YY into the same format returned by the CALENDAR intrinsic.

SYNTAX

```
                    I                                   BA
calen'format = CONVERTDATE(date'string)
```

FUNCTIONAL RETURN

The CONVERTDATE procedure returns an integer value equal to the date contained in the date string. This value is in the same format as the return value from the CALENDAR intrinsic:



PARAMETERS

date'string (Byte array, required)
An array containing a date in the format MM/DD/YY that is to be converted into calendar format.

CONDITION CODES

CCG Not returned.
CCL Invalid Date.
CCE Successful.

SPECIAL CONSIDERATIONS

None.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

CONVERTTIME

PURPOSE

Convert the time in the format HH:MM into the same format returned by the TIMER intrinsic.

SYNTAX

```
                D                                   BA
timer'format = CONVERTTIME(time'string)
```

FUNCTIONAL RETURN

The CONVERTTIME procedure returns a double integer value equal to a millisecond representation of the time contained in the string.

PARAMETERS

time'string (Byte array, required)
An array containing the time in the format HH:MM that is to be converted into millisecond format.

CONDITION CODES

CCG Not returned.
CCL Invalid time.
CCE Successful.

SPECIAL CONSIDERATIONS

None.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

CONVEXTLABELTODELTAP

PURPOSE

Returns the DELTAP of the entry point identified by the STT and segment number contained in the external label.

SYNTAX

```
   I                                          IV
deltap = CONVEXTLABELTODELTAP(extlabel)  O-P
```

FUNCTIONAL RETURN

The returned value is the DELTAP offset to the entry point of the external procedure.

PARAMETERS

extlabel (Integer by value, required)
The external label identifying the segment and STT.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

An invalid external label or one not contained in the calling process's Segment Locality List will cause unpredictable results!

Copyright (C) 1989 Propellerhead Publishing

COMSEARCH

PURPOSE

Returns information about a command from the system command directory.

SYNTAX

```
 L                          BA    IV  D   D    L
valid = COMSEARCH(command, len, cap, access, plabel,
                           I   I
                     error, num) O-P
```

FUNCTIONAL RETURN

A logical value of TRUE indicate successful, FALSE if the command does not exist in the system command directory.

PARAMETERS

command (Byte array, required)
The command to be name to be used for searching the system directory.

len (Integer by value, required)
The length of the command being examined in *command*.

cap (Double, required)
The user capability mask required to use the command.

access (Double, required)
Specific requirements governing the use of the command as follows:

Word	Bits	Description
0	0:10	Unused.
	10:6	Command mask index.
1	0:3	Unused.
	3:1	Command allowed during break.
	4:1	Operator only command.
	5:1	Command not REDOable.
	6:1	Command not allowed in UDC.
	7:1	Execute even if in a FALSE IF statement.

Copyright (C) 1989 Propellerhead Publishing

Word	Bits	Description
1	8:1	Not permitted with character set.
	9:1	Command can be aborted.
	10:1	Batch only Command.
	11:1	Session only command.
	12:2	= 0: No capability check required.
		= 1: All capabilities required.
		= 2: Any one capability required.
	14:1	Not programmatically executable.
	15:1	Not allowed during break.

plabel (Logical, required)
The label for the command execution procedure.

error (Integer, required)
A command interpreter error number.

num (Integer, required)
A number used to identify the command.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

CRYPT

PURPOSE

Encrypts a password.

SYNTAX

```
                    BA    BA
CRYPT(password, seed)
```

PARAMETERS

password (Byte array, required)
An array containing the password to be encrypted. Upon return, this array will contain the encrypted password.

seed (Byte array, required)
The encryption seed generated by the GETSALT procedure.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode. Also, this is a PASCAL procedure and only callable from SPL or PASCAL due to the parameter checking level.

DISCUSSION

See the GETSALT procedure.

Copyright (C) 1989 Propellerhead Publishing

DATEJUL

PURPOSE

Converts gregorian date to julian.

SYNTAX

```
D                          IV    IV    IV
julian = DATEJUL(day, month, year)
```

FUNCTIONAL RETURN

The julian date.

PARAMETERS

day (Integer by value, required)
The day of the month.

month (Integer by value, required)
The month of the year (1 = January, 12=December).

year (Integer by value, required)
The year (i.e. 1988).

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

None.

DISCUSSION

Using a julian date provides the ability to add or subtract dates without having to deal with leap years. See the procedure JULDATE.

Copyright (C) 1989 Propellerhead Publishing

DEALLORIN

PURPOSE

Releases a RIN and returns it to the list of available RINs.

SYNTAX

```
                    IV    LA
DEALLORIN(rin, username)  O-P, O-V
```

PARAMETERS

rin (Integer by value, required)
The RIN number to be released. If bit 0 of this parameter is on, it indicates the calling process's DB register is pointing to the RIN table (local and file RINS only).

username (Logical array, optional)
An array containing the name of the user and account that owns the RIN. This is required if the RIN being released was a global RIN.

CONDITION CODES

CCL The specified RIN was not allocated.
CCE Successful.
CCG The specified RIN is currently being used.

SPECIAL CONSIDERATIONS

Requires privilege mode.

DISCUSSION

This procedure compliments the ALLORIN procedure.

Copyright (C) 1989 Propellerhead Publishing

DEBLANK

PURPOSE

Locate the last non-blank character in a byte array.

SYNTAX

```
I                          BA    IV
len = DEBLANK(buffer, bufferlen)
```

FUNCTIONAL RETURN

The DEBLANK procedure returns an integer value representing the length of the data contained in the buffer with respect to the specified buffer address.

PARAMETERS

buffer (Byte array, required)
An array containing the string whose length is to be determined.

bufferlen (Integer by value, required)
A number indicating the maximum length of the buffer in bytes.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

None.

DISCUSSION

None.

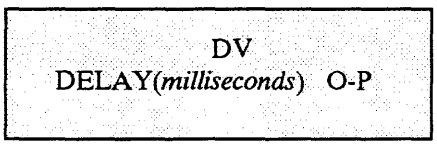
Copyright (C) 1989 Propellerhead Publishing

DELAY

PURPOSE

Causes the calling process to suspend for the specified amount of time.

SYNTAX



PARAMETERS

milliseconds (Double by value, required)
The amount of time, in milliseconds, for the calling process to suspend.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

When using this procedure to suspend a process, the process will resume at the same level in its current queue. The standard PAUSE intrinsic causes the process to be placed back at the top of its executing queue. While the process is in DELAY the impede flag will be set in the PCB and the TRL entry will point to the PCB.

Copyright (C) 1989 Propellerhead Publishing

DELETEDEVFILE

PURPOSE

Delete input/output spoolfiles.

SYNTAX

```
                                    I                    IV  LV
error = DELETEDEVFILE(dfid, byjob)  O-P
```

FUNCTIONAL RETURN

The returned error code has the following values:

Value	Meaning
0	Successful.
2	Invalid/Non-existent spoolfile.
3	Spoolfile not available.

PARAMETERS

dfid (Integer by value, required)
This parameter contains the ID of the spoolfile to be deleted. The contents of this parameter depend on the *byjob* flag:

Byjob		Dfid
0	(0:1) = 0	dfid is an input spoolfile number.
	(0:1) = 1	dfid is an output spoolfile number.
1	(0:2) = 0	dfid is session number from a SPOOK tape.
	(0:2) = 1	dfid is a normal session number.
	(0:2) = 2	dfid is a job number from a SPOOK tape.
	(0:2) = 3	dfid is a normal job number.

Copyright (C) 1989 Propellerhead Publishing

byjob (Logical by value, required)
This is a flag to indicate whether the *dfid* parameter contains a spoolfile ID number or a job/session number. (see 'dfid')

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

None.

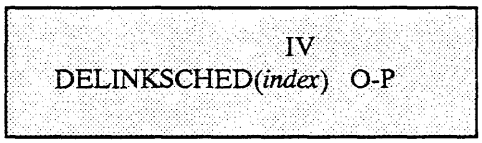
Copyright (C) 1989 Propellerhead Publishing

DELINKSCHED

PURPOSE

Removes a entry from the scheduled job queue.

SYNTAX



PARAMETERS

index (Integer by value, required)
The address in the JMAT table of the job which is to be removed from the schedule queue.

CONDITION CODES

Not returned.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

Must be called with DB pointing to the JMAT table. Also, the JMAT SIR must be locked/unlocked by the calling process.

Copyright (C) 1989 Propellerhead Publishing

DEVSPEC

PURPOSE

Provides the actual device class name associated with a device class index.

SYNTAX

```
                I                    IV    BA
len = DEVSPEC(index, class)  O-P
```

FUNCTIONAL RETURN

Returns the length, in bytes, of the device class name contained in the *class* parameter.

PARAMETERS

index (Integer by value, required)
If negative, represents a device class index. If positive, contains an logical device number.

class (Byte array, required)
If a negative device class index was used, this contains the device class name. When a positive device number is used, the character representation of that number is returned.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

Device class indexes are primarily found in the XDD tables.

Copyright (C) 1989 Propellerhead Publishing

DIRECADJUST

PURPOSE

To adjust the group and account file space counts.

SYNTAX

```
                                    D                    DV      IV      LA
status = DIRECADJUST(numsects, dummy, acct,
                                    LA    IV
                          group, volume)  O-P O-V
```

FUNCTIONAL RETURN

A zero value indicates successful completion. Otherwise the two words contain the following to indicate the error encountered:

NOTE ->	Word 1	Word 0	Error
	1	0	Duplicate entry exists.
	2	n	n=0 - Non-existent file, n=1 - Non-existent group, n=2 - Non-existent account, n=3 - Non-existent user, n=4 - Non-existent volume.
	3	n	n=1 - User cannot access group, n=2 - User cannot access account.
	4	n	No more room, subtree cannot hold any more entry blocks; n% of total.
	5	0	No more room, more than 64K in entries.
	6	n	No more room, directory cannot hold n contiguous blocks.
	7	0	Purge failed, entry was in use.
	8	n	File space limit exceeded for: n=1 - group, n=2 - account.

Copyright (C) 1989 Propellerhead Publishing

DIRECADJUST (continued)

PARAMETERS

numsects (Double by value, required)
Number of sectors by which the account and group file sector counts is to be adjusted. A negative number indicates the file space counts are to be decremented, a positive number indicates the counts are to be incremented.

dummy (Integer by value)
Dummy parameter included for compatibility. Not used.

acct (Array, required)
A logical array containing the name of the account to be searched.

group (Array, required)
An array containing the group or user name, depending upon the value in *type*.

volume (Integer by value, required)
The mounted private volume table index, zero if searching the system file domain.

CONDITION CODES

CCG Error occurred, check functional return.
CCL Not returned.
CCE Successful.

SPECIAL CONSIDERATIONS

Requires privileged mode.
Arithmetic traps must be disabled while calling this procedure.
DIRECINSERTFILE and DIRECPURGEFILE perform the file space adjustment automatically

DISCUSSION

See Expansions Section for full details of calling directory procedures.

Copyright (C) 1989 Propellerhead Publishing

DIRECFIND

PURPOSE

Obtain entries from the system directory.

SYNTAX

```
 D                              IV  DV LA  LA   LA
status = DIRECFIND(type, link, acct, group, file,
                                LA
                   ntry)        O-P
```

FUNCTIONAL RETURN

A zero value indicates successful completion. Otherwise the two words contain the following to indicate the error encountered:

NOTE ->	Word 1	Word 0	Error
	1	0	Duplicate entry exists.
	2	n	n=0 - Non-existent file, n=1 - Non-existent group, n=2 - Non-existent account, n=3 - Non-existent user, n=4 - Non-existent volume.
	3	n	n=1 - User cannot access group, n=2 - User cannot access account.
	4	n	No more room, subtree cannot hold any more entry blocks; n% of total.
	5	0	No more room, more than 64K in entries.
	6	n	No more room, directory cannot hold n contiguous blocks.
	7	0	Purge failed, entry was in use.
	8	n	File space limit exceeded for: n=1 - group, n=2 - account.

PARAMETERS

type (Integer by value, required)
Indicates the type of search to be performed as follows:

Copyright (C) 1989 Propellerhead Publishing

Bits	Action
13:3	Type of search to be done: 0 -- Global, start at root 1 -- Logon account, start at directory block for the logon account, 'vtabx' contains the account group index block (located in JIT). 2 -- Logon group, start at directory block for the logon group, 'vtabx' contains the group file index block (located in JIT).
10:3	Type of entry to be located: 0 -- File entry, 1 -- Group entry, 2 -- Account entry, 3 -- User entry, 4 -- Volume entry.

vtabx (Double by value, required);
Volume table index pointer constructed as follows:

Word	Description
0	Volume table index of the mounted private volume. 0 if searching system domain.
1	The directory index pointer where the search is to start (when type.(13:3) <> 0). This may be obtained from the JIT or by the system procedure PRODUCEPARMS.

acct (Array, required)
A 4-word array containing the account name to be located. The name must be left-justified in the array and padded with trailing blanks.

group (Array, required)
A 4-word array containing the group name to be retrieved for type.(10:3) = 0, 1, or 4. If search type.(10:3) = 3, this array contains the user name. In either case, the name is left-justified and padded with trailing blanks. If the search type.(10:3) = 2 (account search), this argument is ignored.

file (Array, required)
A 4-word array containing the file name to be located if the search type.(10:3) = 0. If the search type.(10:3) = 4, this array contains the volume set name. The name is left-justified and padded with trailing blanks. If the search type.(10:3) <> 0 or 4 (file or volume search), this argument is ignored.

ntry (Array, required)
This contains the directory entry found. The size of the array depends on the search performed:

Copyright (C) 1989 Propellerhead Publishing

Search	Size
File	6
Group	41
Account	30
User	19
Volume	56

CONDITION CODES

CCG Error occurred, check functional return.
CCL Not returned.
CCE Successful.

SPECIAL CONSIDERATIONS

Requires privileged mode. If used incorrectly, this procedure could cause a system failure. As with all directory procedures, disable arithmetic traps prior to calling.

DISCUSSION

Refer to Chapter 4 of the MPE-V System Tables Manual for formats of entries returned.

Copyright (C) 1989 Propellerhead Publishing

DIRECFINDFILE

PURPOSE

Obtain file entries from the system directory.

SYNTAX

```
                                    D                                IV   DV LA  LA
status = DIRECFINDFILE(type, link, acct, group,
                                                                    LA LA      IV
                                                    file, freturn, mvtabx)  O-P O-V
```

FUNCTIONAL RETURN

A zero value indicates successful completion. Otherwise the two words contain the following to indicate the error encountered:

NOTE ->	Word 1	Word 0	Error
	1	0	Duplicate entry exists.
	2	n	n=0 - Non-existent file, n=1 - Non-existent group, n=2 - Non-existent account, n=3 - Non-existent user, n=4 - Non-existent volume.
	3	n	n=1 - User cannot access group, n=2 - User cannot access account.
	4	n	No more room, subtree cannot hold any more entry blocks; n% of total.
	5	0	No more room, more than 64K in entries.
	6	n	No more room, directory cannot hold n contiguous blocks.
	7	0	Purge failed, entry was in use.
	8	n	File space limit exceeded for: n=1 - group, n=2 - account.

PARAMETERS

type (Integer by value, required)
Indicates the type of search to be performed as follows:

Copyright (C) 1989 Propellerhead Publishing

Bits	Action
13:3	Type of search to be done: 0 -- Global, start at root 1 -- Logon account, start at directory block for the logon account, 'vtabx' contains the account group index block (located in JIT). 2 -- Logon group, start at directory block for the logon group, 'vtabx' contains the group file index block (located in JIT).
10:3	Type of entry to be located: 0 -- File entry, 1 -- Group entry, 2 -- Account entry, 3 -- User entry, 4 -- Volume entry.

link (Double by value, required);
Volume table index pointer constructed as follows:

Word	Description
0	Volume table index of the mounted private volume. 0 if searching system domain.
1	The directory index pointer where the search is to start (when type.(13:3) <> 0). This may be obtained from the JIT or by the system procedure PRODUCEPARMS.

acct (Array, required)
A 4-word array containing the account name to be located. The name must be left-justified in the array and padded with trailing blanks.

group (Array, required)
A 4-word array containing the group name to be retrieved.

file (Array, required)
A 4-word array containing the file name to be located,

freturn (Array, required)
This contains an array of data, containing the file pointer and the security masks from the group and/or account. The format is as follows:

Word	Bits	Description
0	0:8	Volume Table Index indicating disc on which file label is stored.
0	8:8	High Order Disc Address of file label.
1	0:16	Low Order Disc Address of file label.
2-3	All	The group security mask.
4	0:16	The account security mask, if the search level in the *type* parameter is 0 or 1.

mvtabx (Integer by value, optional)

Copyright (C) 1989 Propellerhead Publishing

The mounted volume table index, indicating the private volume set to search. This number is returned by the MOUNT procedure. Refer to the description of MOUNT for more details.

CONDITION CODES

CCG Error occurred, check functional return.
CCL Not returned.
CCE Successful.

SPECIAL CONSIDERATIONS

Requires privileged mode. If used incorrectly, this procedure could cause a system failure. As with all directory procedures, disable arithmetic traps prior to calling.

DISCUSSION

This procedure only returns the 6 word directory file entry, not the file label itself. The label must be retrieved by decoding the last 2 words returned, and using FLABIO or ATTACHIO.

Copyright (C) 1989 Propellerhead Publishing

DIRECINSERT

PURPOSE

To insert a group, account, user or volume entry into the directory and adjust the appropriate internal pointers.

SYNTAX

```
D                                IV   DV LA  LA
status = DIRECINSERT(type, link, acct, group,
                                 LA LA    IV
                     file, ntry, mvtabx)  O-P O-V
```

FUNCTIONAL RETURN

A zero value indicates successful completion. Otherwise the two words contain the following to indicate the error encountered:

NOTE ->	Word 1	Word 0	Error
	1	0	Duplicate entry exists.
	2	n	n=0 - Non-existent file, n=1 - Non-existent group, n=2 - Non-existent account, n=3 - Non-existent user, n=4 - Non-existent volume.
	3	n	n=1 - User cannot access group, n=2 - User cannot access account.
	4	n	No more room, subtree cannot hold any more entry blocks; n% of total.
	5	0	No more room, more than 64K in entries.
	6	n	No more room, directory cannot hold n contiguous blocks.
	7	0	Purge failed, entry was in use.
	8	n	File space limit exceeded for: n=1 - group, n=2 - account.

Copyright (C) 1989 Propellerhead Publishing

DIRECINSERT (continued)

PARAMETERS

type (Integer by value, required)
Indicates the type of search to be performed as follows:

Bits	Action
13:3	Search STARTLEVEL: 0 -- Global, start at root 1 -- Logon account, start at directory block for the logon account, 'vtabx' contains the account group index block (located in JIT). 2 -- Logon group, start at directory block for the logon group, 'vtabx' contains the group file index block (located in JIT).
10:3	Search ENDLEVEL: 0 -- File entry, 1 -- Group entry, 2 -- Account entry, 3 -- User entry, 4 -- Volume entry.

link (Double by value, required);
Directory address where search is to start. Required is STARTLEVEL is anything other than zero (0).

acct (Array, required)
A logical array containing the name of the account to be searched.

group (Array, required)
An array containing the group or user name, depending upon the value in *type*.

file (Array, required)
An array containing the file name. Used with *type* indicates a volume set entry is being inserted

ntry (Array, required)
A pointer to the word **after** the name within the formatted directory entry to be inserted. This would point to the fourth word of the group or account entry, which would be the file or group index pointer, respectively.

mvtabx (Integer by value, required)
The mounted private volume table index, zero if searching the system file domain.

Copyright (C) 1989 Propellerhead Publishing

DIRECINSERT (continued)

CONDITION CODES

CCG Error occurred, check functional return.
CCL Not returned.
CCE Successful.

SPECIAL CONSIDERATIONS

Requires privileged mode.
Arithmetic traps must be disabled while calling this procedure.
File entries are inserted by the procedure DIRECINSERTFILE.

DISCUSSION

See Expansions Section for full details of calling directory procedures.

Copyright (C) 1989 Propellerhead Publishing

DIRECINSERTFILE

PURPOSE

To insert a file entry into the directory and adjust the group and account sector counts.

SYNTAX

```
                                                    DV     IV     LA   LA
status = DIRECINSERTFILE(numsects, dummy, acct, group,
                                                LA DV     IV
                                        file, faddr, mvtabx) O-P O-V
```

FUNCTIONAL RETURN

A zero value indicates successful completion. Otherwise the two words contain the following to indicate the error encountered:

NOTE ->	Word 1	Word 0	Error
	1	0	Duplicate entry exists.
	2	n	n=0 - Non-existent file, n=1 - Non-existent group, n=2 - Non-existent account, n=3 - Non-existent user, n=4 - Non-existent volume.
	3	n	n=1 - User cannot access group, n=2 - User cannot access account.
	4	n	No more room, subtree cannot hold any more entry blocks; n% of total.
	5	0	No more room, more than 64K in entries.
	6	n	No more room, directory cannot hold n contiguous blocks.
	7	0	Purge failed, entry was in use.
	8	n	File space limit exceeded for: n=1 - group, n=2 - account.

Copyright (C) 1989 Propellerhead Publishing

DIRECINSERTFILE(continued)

PARAMETERS

type (Integer by value, required)
Indicates the type of search to be performed as follows:

Bits	Action
13:3	Search STARTLEVEL: 0 -- Global, start at root 1 -- Logon account, start at directory block for the logon account, 'vtabx' contains the account group index block (located in JIT). 2 -- Logon group, start at directory block for the logon group, 'vtabx' contains the group file index block (located in JIT).
10:3	Search ENDLEVEL: 0 -- File entry, 1 -- Group entry, 2 -- Account entry, 3 -- User entry, 4 -- Volume entry.

numsects (Double by value, required);
Number of sectors that this file occupies. This number will be used when adjusting the account and group filespace counts.

dummy (Integer by value, required)
Required for compatibility reasons. Not used.

acct (Array, required)
A logical array containing the name of the account to be searched.

group (Array, required)
An array containing the group name, depending upon the value in *type*.

file (Array, required)
An array containing the file name.

faddr (Array, required)
The double-word disc address of the file label, which will be used to complete the file entry in the directory. *Faddr* has the following format:

Word	Bits	Description
0	0:8	Volume table index for disc drive on which the file label resides

Copyright (C) 1989 Propellerhead Publishing

0	8:8	High Order Disc Address (HODA). First 8 bits of disc address
1	0:16	Low Order Disc Address (LODA). The remaining 16 bits. The HODA and LODA actually constitute a 24 bit disc address.

mvtabx (Integer by value, required)
The mounted private volume table index, zero if searching the system file domain.

CONDITION CODES

CCG Error occurred, check functional return.
CCL Not returned.
CCE Successful.

SPECIAL CONSIDERATIONS

DIRECINSERTFILE always checks to ensure the caller has save file (SF) access to the file's group.
Requires privileged mode.
Arithmetic traps must be disabled while calling this procedure.
Other types of directory entries are inserted by the procedure DIRECINSERT.
The entry point DIRECRESETFILE functions identically, but bypasses the security check.

DISCUSSION

See Expansions Section for full details of calling directory procedures.

Copyright (C) 1989 Propellerhead Publishing

DIRECLOGOFF

PURPOSE

Updates the system directory (counters) along with cpu and connect time for the logging-off user.

SYNTAX

```
                                    IV  LA   DV   DV   LA
I
result = DIRECLOGOFF(mask, jmat, connect, cpu, account,
                                    LA  LA
                     user, group) O-P
```

FUNCTIONAL RETURN

The integer value returned from the procedure is as follows:

Bits	Meaning
15:1	Account connect time exceeded
14:1	Account cpu time exceeded
13:1	Group connect time exceeded
12:1	Group cpu time exceeded
11:1	Account connect time negative
10:1	Group connect time negative
9:1	Account cpu time negative
8:1	Group cpu time negative

PARAMETERS

mask (Integer by value, required)
The integer value indicating the following:

Value	Meaning
0	Account, group and user all exist
1	Group does not exist.
2	Account does not exist.
3	User does not exist.
4	No home group specified.

Copyright (C) 1989 Propellerhead Publishing

DIRECLOGOFF (continued)

jmat (Logical array, required)
The complete JMAT entry which is used to locate the account, group and user directory entries to be updated.

connect (Double by value, required)
The amount of connect time to increment the group and account.

cpu (Double by value, required)
The amount of cpu time to increment the group and account.

account (Logical array, required)
Not used.

user (Logical array, required)
Not used.

group (Logical array, required)
Not used.

CONDITION CODES

Unknown.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

As with all directory procedures, disable arithmetic traps prior to calling.

Copyright (C) 1989 Propellerhead Publishing

DIRECLOGON

PURPOSE

Updates the system directory (counters) to reflect the new logon user.

SYNTAX

```
                I                          IV  LA   DV   DV   LA
result = DIRECLOGON(mask, jmat, connect, cpu, account,
                                           LA  LA
                                    user, group)  O-P
```

FUNCTIONAL RETURN

The integer value returned from the procedure is as follows:

Value	Meaning
0	Successful.
1	Group does not exist.
2	Account does not exist.
3	User does not exist.
4	No home group specified.

PARAMETERS

mask (Integer by value, required)
Must be zero (0).

jmat (Logical array, required)
The complete JMAT entry which is used to retrieve the account, group and user directory entries.

connect (Double by value, required)
Ignored, specify zero (0).

cpu (Double by value, required)
Ignored, specify zero (0).

account (Logical array, required)
The account directory entry returned.

user (Logical array, required)
The user directory entry returned.

Copyright (C) 1989 Propellerhead Publishing

group (Logical array, required)
The group directory entry returned.

CONDITION CODES

Unknown.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

As with all directory procedures, disable arithmetic traps prior to calling.

Copyright (C) 1989 Propellerhead Publishing

DIRECPURGE

PURPOSE

To purge an account, group, user, or volume entry from the directory.

SYNTAX

```
 D                                IV  DV  LA  LA
status = DIRECPURGE(type, link, acct, group,
                                  LA  IV
                    file, mvtabx)  O-P O-V
```

FUNCTIONAL RETURN

A zero value indicates successful completion. Otherwise the two words contain the following to indicate the error encountered:

NOTE ->	Word 1	Word 0	Error
	1	0	Duplicate entry exists.
	2	n	n=0 - Non-existent file, n=1 - Non-existent group, n=2 - Non-existent account, n=3 - Non-existent user, n=4 - Non-existent volume.
	3	n	n=1 - User cannot access group, n=2 - User cannot access account.
	4	n	No more room, subtree cannot hold any more entry blocks; n% of total.
	5	0	No more room, more than 64K in entries.
	6	n	No more room, directory cannot hold n contiguous blocks.
	7	0	Purge failed, entry was in use.
	8	n	File space limit exceeded for:

Copyright (C) 1989 Propellerhead Publishing

```
                                    n=1 - group, n=2 -
                                    account.
```

PARAMETERS

type (Integer by value, required)
Indicates the type of search to be performed as follows:

Bits	Action
13:3	Type of search to be done: 0 -- Global, start at root 1 -- Logon account, start at directory block for the logon account, 'vtabx' contains the account group index block (located in JIT). 2 -- Logon group, start at directory block for the logon group, 'vtabx' contains the group file index block (located in JIT).
10:3	Type of entry to be located: 0 -- File entry, 1 -- Group entry, 2 -- Account entry, 3 -- User entry, 4 -- Volume entry.

link (Double by value, required);
Directory address where search is to start. Required is STARTLEVEL is anything other than zero (0).

acct (Array, required)
A logical array containing the name of the account to be searched.

group (Array, required)
An array containing the group or user name, depending upon the *type* parameter (ie, search being performed).

file (Array, required)
An array containing the file name.

mvtabx (Integer by value, required)
The mounted private volume table index, zero if searching the system file domain.

CONDITION CODES

CCG Error occurred, check functional return.
CCL Not returned.
CCE Successful.

Copyright (C) 1989 Propellerhead Publishing

DIRECPURGE(continued)

SPECIAL CONSIDERATIONS

Requires privileged mode.
Arithmetic traps must be disabled while calling this procedure.

DISCUSSION

File entries are removed by the procedure DIRECPURGEFILE.
See Expansions section for discussion of this and other directory procedures.

Copyright (C) 1989 Propellerhead Publishing

DIRECPURGEFILE

PURPOSE

To purge a file entry from the directory and adjust the group and account file space counts.

SYNTAX

```
 D                                    DV     IV     LA
status = DIRECPURGEFILE(numsects, dummy, acct,
                                      LA   LA   IV
                        group, file, mvtabx)  O-P O-V
```

FUNCTIONAL RETURN

A zero value indicates successful completion. Otherwise the two words contain the following to indicate the error encountered:

NOTE ->	Word 1	Word 0	Error
	1	0	Duplicate entry exists.
	2	n	n=0 - Non-existent file, n=1 - Non-existent group, n=2 - Non-existent account, n=3 - Non-existent user, n=4 - Non-existent volume.
	3	n	n=1 - User cannot access group, n=2 - User cannot access account.
	4	n	No more room, subtree cannot hold any more entry blocks; n% of total.
	5	0	No more room, more than 64K in entries.
	6	n	No more room, directory cannot hold n contiguous blocks.
	7	0	Purge failed, entry was in use.
	8	n	File space limit exceeded for:

Copyright (C) 1989 Propellerhead Publishing

DIRECPURGEFILE(continued)

```
                                    n=1 - group, n=2 -
                                    account.
```

PARAMETERS

numsects (Double by value, required);
Number of sectors to adjust the group and account file space counts. Negative numbers cause the value to be decremented, positive to increment.

dummy (Integer by value)
Included for compatibility reasons. Not used.

acct (Array, required)
A logical array containing the name of the account to be searched.

group (Array, required)
An array containing the group or user name, depending upon the *type* parameter (ie, search being performed).

file (Array, required)
An array containing the file name.

mvtabx (Integer by value, required)
The mounted private volume table index, zero if searching the system file domain.

CONDITION CODES

CCG Not returned.
CCL Not returned.
CCE Successful.

SPECIAL CONSIDERATIONS

Requires privileged mode.
Arithmetic traps must be disabled while calling this procedure.

DISCUSSION

Other entries are removed by the procedure DIRECPURGE.
See Expansions section for discussion of this and other directory procedures.

Copyright (C) 1989 Propellerhead Publishing

DIRECSCAN

PURPOSE

Locate entries in the system directory.

SYNTAX

```
                D                          IV   DV  LA  LA   LA
status = DIRECSCAN(type, link, acct, group, file,
                                        IP    LA      IV
                                   recip, parms, mvtabx)  O-P
```

FUNCTIONAL RETURN

A zero value indicates successful completion. Otherwise the two words contain the following to indicate the error encountered:

NOTE ->	Word 1	Word 0	Error
	1	0	Duplicate entry exists.
	2	n	n=0 - Non-existent file, n=1 - Non-existent group, n=2 - Non-existent account, n=3 - Non-existent user, n=4 - Non-existent volume.
	3	n	n=1 - User cannot access group, n=2 - User cannot access account.
	4	n	No more room, subtree cannot hold any more entry blocks; n% of total.
	5	0	No more room, more than 64K in entries.
	6	n	No more room, directory cannot hold n contiguous blocks.
	7	0	Purge failed, entry was in use.
	8	n	File space limit exceeded for: n=1 - group, n=2 - account.

Copyright (C) 1989 Propellerhead Publishing

DIRECSCAN(continued)

PARAMETERS

type (Integer by value, required)
Indicates the type of search to be performed as follows:

Bits	Action
13:3	Search STARTLEVEL: 0 -- Global, start at root 1 -- Logon account, start at directory block for the logon account, 'vtabx' contains the account group index block (located in JIT). 2 -- Logon group, start at directory block for the logon group, 'vtabx' contains the group file index block (located in JIT).
10:3	Search ENDLEVEL: 0 -- File entry, 1 -- Group entry, 2 -- Account entry, 3 -- User entry, 4 -- Volume entry.
9:1	ALLFLAG. Flag to visit all of ENDLEVEL.
6:3	Ultimate TOLEVEL scan is to achieve. Options same as ENDLEVEL.
5:1	HITFLAG. Visit every node 'hit'.

link (Double by value, required);
Directory address where search is to start. Required is STARTLEVEL is anything other than zero (0).

acct (Array, required)
A logical array containing the name of the account to be searched.

group (Array, required)
An array containing the group or user name, depending upon the *type* parameter (ie, search being performed).

file (Array, required)
An array containing the file name.

recip (Integer procedure, required)
The plabel of the procedure to be called once a directory entry matching the qualifications is located. The arguments for the procedure are as follows:

Copyright (C) 1989 Propellerhead Publishing

```
          I                LA     IV     IV    DV
          i'ret := RECIP(ntry, level, parms, sirs)

      Where: i'ret  -   return value to DIRECSCAN as
                        follows:

                15:1 = 0 ->Did a release SIR
                       1 ->No release SIR
                13:2 = 0 ->Continue with search
                       1 ->Skip current subtree
                       2 ->Stop  scan  of  parent's
                           subtree

             ntry  -    DB-relative pointer to entry in
                        DIR seg.

             level -    The directory level.

             parms -    Address of PARMS array passed
                        to  DIRECSCAN,  relative  to
                        DIRECSCAN's (RECIP's caller) Q
                        register. See Expansion section
                        for more details.

             getsir -   Double word value containing the
                        SIR number in word 0 and the
                        value returned by GETSIR in word
                        1.
```

Special Notes:

- Directory SIR is double-locked at the intermediate levels.
- RECIP must not re-acquire the Directory SIR if released.
- RECIP must release all resources prior to exit.
- DB must be pointing to the directory segment upon exit.

parms (Array, required)
An array containing the search parameters which is formatted by the PRODUCEPARMS procedure.

mvtabx (Integer by value, required)
The mounted private volume table index, zero if searching the system file domain.

CONDITION CODES

CCG Error occurred, check functional return.
CCL Not returned.
CCE Successful.

Copyright (C) 1989 Propellerhead Publishing

DIRECSCAN(continued)

SPECIAL CONSIDERATIONS

Requires privileged mode.
Arithmetic traps must be disabled while calling this procedure.

DISCUSSION

The PRODUCEPARMS procedure facilitates constructing the parameters for this procedure, including *type*. The procedure DIRMATCH may be useful in the *recip* procedure to determine if a match is found. See Expansions Section for full details of calling directory procedures.

Copyright (C) 1989 Propellerhead Publishing

DIREC[RE]SETFLAG

PURPOSE

To reset/set the Bad File Label flag in a file entry in the directory.

SYNTAX

```
 D                                  IV  DV LA  LA
status = DIRECRESETFLAG(type, link, acct, group,
                                    LA  IV
                        file, mvtabx) O-P O-V

 D                                 IV  DV LA  LA
status = DIRECSETFLAG(type, link, acct, group,
                                   LA  IV
                       file, mvtabx) O-P O-V
```

FUNCTIONAL RETURN

A zero value indicates successful completion. Otherwise the two words contain the following to indicate the error encountered:

NOTE ->	Word 1	Word 0	Error
	1	0	Duplicate entry exists.
	2	n	n=0 - Non-existent file, n=1 - Non-existent group, n=2 - Non-existent account, n=3 - Non-existent user, n=4 - Non-existent volume.
	3	n	n=1 - User cannot access group, n=2 - User cannot access account.
	4	n	No more room, subtree cannot hold any more entry blocks; n% of total.
	5	0	No more room, more than 64K in entries.
	6	n	No more room, directory cannot hold n contiguous blocks.

Copyright (C) 1989 Propellerhead Publishing

7	0	Purge failed, entry was in use.
8	n	File space limit exceeded for: n=1 - group, n=2 - account.

PARAMETERS

type (Integer by value, required)
Indicates the type of search to be performed as follows:

Bits	Action
13:3	Type of search to be done: 0 -- Global, start at root 1 -- Logon account, start at directory block for the logon account, 'vtabx' contains the account group index block (located in JIT). 2 -- Logon group, start at directory block for the logon group, 'vtabx' contains the group file index block (located in JIT).
10:3	Type of entry to be located: 0 -- File entry, 1 -- Group entry, 2 -- Account entry, 3 -- User entry, 4 -- Volume entry.

link (Double by value, required);
Directory address where search is to start. Required is STARTLEVEL is anything other than zero (0).

acct (Array, required)
A logical array containing the name of the account to be searched.

group (Array, required)
An array containing the group or user name, depending upon the *type* parameter (ie, search being performed).

file (Array, required)
An array containing the file name.

mvtabx (Integer by value, required)
The mounted private volume table index, zero if searching the system file domain.

CONDITION CODES

Copyright (C) 1989 Propellerhead Publishing

CCG File Entry already flagged.
CCL File not found.
CCE File entry flagged.

SPECIAL CONSIDERATIONS

Requires privileged mode.
Arithmetic traps must be disabled while calling this procedure.

DISCUSSION

See Expansions section for discussion of this and other directory procedures.

Copyright (C) 1989 Propellerhead Publishing

DIRMATCH

PURPOSE

Determines if the actual account structure entry name matches the designated name.

SYNTAX

```
                I                       BA      BA
match = DIRMATCH(designator, real'name)  O-P
```

FUNCTIONAL RETURN

The returned error code has the following values:

Value	Meaning
-1	Designator is less than real name
0	Match
1	Designator is greater than real name

PARAMETERS

designator (Byte array, required)
Contains the generic name with wildcard characters (ABC@, etc).

real'name (Byte array, required)
Actual name of the entry to be tested.

CONDITION CODES

Not returned.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

This procedure is useful in any procedure passed to DIRECSCAN to determine if a directory entry qualifies in the search. See the Expansions section for a further discussion.

Copyright (C) 1989 Propellerhead Publishing

DISCSIZE

PURPOSE

Return the maximum size (address) of a particular logical disc device.

SYNTAX

```
 D                    IV
size = DISCSIZE(ldev)  O-P
```

FUNCTIONAL RETURN

The returned value is the maximum (largest) addressable sector on the specified disc device.

PARAMETERS

ldev (Integer by value, required)
The logical device of the disc drive. Note, this is not the volume table index.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

This procedure can be used to insure a given disc address is within the addressable area on a disc device. Most procedures requiring disc addresses as arguments will cause system failures should they receive a bad sector address.

Copyright (C) 1989 Propellerhead Publishing

DISCTYPE

PURPOSE

Given an disc LDEV, returns the disc volume label along with a number indicating the disc drive model.

SYNTAX

```
 I                    IV  LA
type = DISCTYPE(ldev, label)  O-P
```

FUNCTIONAL RETURN

A number identifying the disc model number as follows:

Value	Disc Model
1	(Unknown).
2	7920
3	7933/7935
4	7945
5	7937

PARAMETERS

ldev (Integer by value, required)
The logical device number of the disc.

label (Logical array, required)
The disc volume label from the specified device. Must be at least 128 words.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

DISMOUNT

PURPOSE

To terminate a logical connection (mount) of a Private Volume set and remove any binding information that may have been established by the corresponding MOUNT.

SYNTAX

```
                    LA      LA     LA     I
DISMOUNT (vsname, vsgroup, vsaccnt, reqtype,

                    IV     I        IV
                gen, pvinfo, different'pin) O-V, O-P
```

PARAMETERS

vsname Specifies the volume set volume name to be mounted. For example, **PVSYS**.PUB.SYS. Asterisk (*) denotes home volume set of user.

vsgroup Specifies the volume set group name to be mounted. For example, PVSYS.**PUB**.SYS.

vsaccnt Specifies the volume set account name to be mounted. For example, PVSYS.PUB.**SYS**.

reqtype An integer code specifying the type of mount to be performed. Generally, the same request code will be used on both the MOUNT and DISMOUNT calls. The different request codes, actions, and examples of users are as follows:

Code	Action(User)
-3	System Dismount (Conditional), Binding (PVCLOSE, IMPLICIT'MNT)
-2	System Dismount (Conditional), No Binding
-1	Unused
0	User Dismount, Binding (CXDISMOUNT if * specified)
1	User Dismount, No Binding (CXDISMOUNT if other than * specified)
2	System Dismount, Binding (FCLOSE)
3	System Dismount, No Binding (NEW/ALT/PURGEGROUP ALT/PURGEACCT REPORT)
4	Operator Dismount, No Binding (LDISMOUNT)

gen Generation number of volume set.

Copyright (C) 1989 Propellerhead Publishing

DISMOUNT

pvinfo Private volume information returned by MOUNT when the volume set is mounted and possibly bound successfully. EITHER *pvinfo* is passed to DISMOUNT or the *vsname*, *vsgroup*, and *vsaccnt*. The format of *pvinfo* is as follows:

```
PVINFO.(0:8)      Volume Mask, indicating volumes of
                  set mounted.
PVINFO.(8:8)      Index into the Mounted Volume Table.
                  This number is passed on to other
                  procedures, such as DIRECSCAN.  This
                  can also be passed to the
                  complementary procedure DISMOUNT.
```

different'pin Passed by some system callers to indicate DISMOUNT is being performed on another process's behalf. Zero indicates mount is being done for calling process.

CONDITION CODES

CCE Successful
CCG Not returned
CCL Other Error

SPECIAL CONSIDERATIONS

Requires Privileged Mode.

DISCUSSION

Complements the procedure MOUNT.

Copyright (C) 1989 Propellerhead Publishing

DSMOVE

PURPOSE

Transfers data between the caller's stack and the specified data segment.

SYNTAX

```
                    IV  IV   IV   IV   IV
DSMOVE(func, dst, offset, count, addr)  O-P
```

PARAMETERS

func (Integer by value, required)
A value of zero (0) to read from the data segment, one (1) to write to the data segment.

dst (Integer by value, required)
The DST number of the target data segment.

offset (Integer by value, required)
An offset into the data segment where the read/write is to start.

count (Integer by value, required)
The number of words to transfer.

addr (Integer by value, required)
The address of the array within the caller's stack used to contain the information being transferred.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

No parameter checking is done!

Copyright (C) 1989 Propellerhead Publishing

DSOPEN

PURPOSE

Opens a DSLINE.

SYNTAX

```
I                    BA  LV  I  BAI  I  I  I
flag = DSOPEN(class, status, p3, p4, p5, p6, p7, p8) O-P  O-V
```

FUNCTIONAL RETURN

The number of the DSLINE which is equal to the offset into the AFT table.

PARAMETERS

class (Byte array, required)
The logical device number or device class of the node to open a line.

status (Logical by value, required)
Not fully known. However, using an %102 opens the line without quiet and a %122 is the quiet mode.

p3 - p8 (Integer, optional)
Unknown at this time.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

DST'SIZE

PURPOSE

Determines the current size of the specified data segment.

SYNTAX

FUNCTIONAL RETURN

The size of the data segment in words.

PARAMETERS

dst (Integer by value, required)
The DST number of the data segment to be examined.

CONDITION CODES

CCL Invalid DST number.
CCE Successful.
CCG DST is unassigned.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

Split-stack calls are permitted.

Copyright (C) 1989 Propellerhead Publishing

ERROREXIT

PURPOSE

Part of the mechanism used to pass error abort information between procedures.

SYNTAX

```
                          LV     LV      LV
ERROREXIT(intexit, errorword, parm)  O-P
```

PARAMETERS

intexit (Logical by value, required)
A logical value composed as follows:

Bits	Meaning
0:10	The intrinsic number.
10:6	The number of words of all formal parameters including the option variable mask if utilized.

errorword (Logical by value, required)
A value to be inserted into the error stack in the PCBX. A value of zero (0) indicates exit out of caller's code normally. If non-zero, then bits 0-7 contain a parameter and bits 8-15 an error byte.

parm (Logical by value, required)
Typically a value of zero.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

Copyright (C) 1989 Propellerhead Publishing

ERROREXIT (continued)

DISCUSSION

ERROREXIT is part of a mechanism used by procedures called by intrinsics to either normally return or abort the user. If the user is not to be aborted, then a stack marker to exist back to the caller is constructed. If abort information exists in the PCBX portion of the stack and the final ERROREXIT is reached, than ABORT is called.

See the procedure ERRORON.

Copyright (C) 1989 Propellerhead Publishing

ERRORON

PURPOSE

Part of the mechanism used to pass error abort information between procedures.

SYNTAX

```
ERRORON O-P
```

PARAMETERS

None.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

ERRORON is called by a procedure if it needs to receive abort code information from a procedure it calls. This abort code information is saved in the PCBX area of the calling process's stack. Only when all ERRORON calls have been matched with a corresponding ERROREXIT will the process be allowed to abort.

The procedure sets a bit in the caller's PCB entry to signify the user is entering system code. This causes certain pseudo interrupts and stack overflow aborts to be postponed until the system code bit is reset via a call to ERROREXIT. ERRORON also disables arithmetic traps.
Split-stack calls are permitted. See ERROREXIT.

Copyright (C) 1989 Propellerhead Publishing

EXCHANGEDB

PURPOSE

Changes the process's DB register to point to the base of the specified data segment.

SYNTAX

FUNCTIONAL RETURN

The EXCHANGEDB procedure returns a logical value equal to the DST number of the current DB register.

PARAMETERS

dst (Logical by value, required)
The DST number of the data segment to be placed in the DB register. To restore the DB register to the process's stack, use a zero DST number.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

This is different than the SWITCHDB intrinsic. The DST number replacing the DB register does not have to belong to the calling program. Potential system failures include:

```
611 - DB is pointing to an absolute address (must do a
      RESETDB first) or the DST number is invalid.
625 - Specified DST is unassigned.
```


Copyright (C) 1989 Propellerhead Publishing

FALTSEC

PURPOSE

Obtains/changes the security matrix and/or security/release state of a file.

SYNTAX

```
                    LV      L       LA      L
FALTSEC(fnum, newsecflag, newmatrix, oldsecflag,
                    LA      L
        oldmatrix, excond) O-V
```

PARAMETERS

fnum (Logical by value, required)
The file number assigned to the file when it was opened.

newsecflag (Logical, optional)
A value of 1 to secure the file, 0 to release it.

newmatrix (Logical Array, optional)
A 2-word array containing the new security matrix to assign to the file.

oldsecflag (Logical, optional)
The current setting of the secured/released flag. A value of zero (0) indicates the file is released, a value of one (1) indicates the file is secured.

oldmatrix (Logical Array, optional)
A 2-word array containing the current security matrix assigned to the file.

excond (Logical, required)
Exceptional conditions which are only warnings:

Value	Meaning
1	'secure' requested, but file already secured.
2	'release' requested, but file already released.
3	Warning: file released but new security matrix applied anyway.
4	New security matrix contains all zeros, so security is unchanged.
5	'Default' and 'Creator-Only' requested, set to 'Creator-Only'.

Copyright (C) 1989 Propellerhead Publishing

FALTSEC(continued)

CONDITION CODES

CCL File system error (use FCHECK).
CCE Successful.
CCG Exceptional condition occurred but change was made.

SPECIAL CONSIDERATIONS

None.

DISCUSSION

File must be opened exclusively with Read/Write access. User MUST be the file's creator. Privileged Mode is not needed.

Copyright (C) 1989 Propellerhead Publishing

FAMILY

PURPOSE

Returns all direct and indirect sons of the specified process.

SYNTAX

```
 I                    IV      IV
son = FAMILY(start'pin, end'pin)  O-P
```

FUNCTIONAL RETURN

The next son PIN found in this family. The end of the family is indicated when *son* equals *end'pin*.

PARAMETERS

start'pin (Integer by value, required)
The starting PIN whose sons/brothers are to be identified.

end'pin (Integer by value, required)
The last PIN to be located.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

The entire family can be identified by continuing to call this procedure using the *son* as a *start'pin* until *end'pin* is returned.

Copyright (C) 1989 Propellerhead Publishing

FGETLOCKWORD

PURPOSE

Returns the lockword of an open file.

SYNTAX

```
 I                              IV      BA     I
error = FGETLOCKWORD(fnum, lockword, len) O-P
```

FUNCTIONAL RETURN

A value of zero (0) indicates successful, otherwise returns a file system error number.

PARAMETERS

fnum (Integer by value, required)
The file number of the opened file. The file must be opened for exclusive or exclusive read access.

lockword (Byte array, required)
The file's lockword, padded with blanks.

len (Integer, required)
The length of the lockword, 0 if no lockword assigned to the file.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Cannot be called in split-stack.

DISCUSSION

Also see the intrinsic FLABELINFO.

Copyright (C) 1989 Propellerhead Publishing

FINDDEVICES

PURPOSE

Identifies the logical devices number(s) for all devices within the requested class.

SYNTAX

```
                              IV      IA
FINDDEVICES(class, devicenums)  O-P
```

PARAMETERS

class (Integer by value, required)
This is a class code indicating the type of devices that are to be identified as follows:

Class	Device Type
1	Disc
2	Printers
3	Tape Drives
4	Terminals
5	INP Devices
6	DS Lines
7	MRJE Lines
8	HSI Devices
9	SSLC Devices

devicenums (Integer array, required)
This array contains the numbers of the devices within the requested class. The array must be large enough to accommodate the expected list plus 1 word. The first word in the array will contain a count of the number of devices contained in the array starting at word 2.

CONDITION CODES

CCL Failed.
CCE Successful.
CCG Failed.

SPECIAL CONSIDERATIONS

Copyright (C) 1989 Propellerhead Publishing

FINDDEVICES(continued)

Requires privileged mode.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

FLABIO

PURPOSE

Read/write a file label.

SYNTAX

```
   I                IV    DV     IV     A
error = FLABIO(ldev, address, function, label)  O-P
```

FUNCTIONAL RETURN

The following values are returned:

Value	Meaning
0	Successful.
1	Hard error.
2	Soft error.

PARAMETERS

ldev (Integer by value, required)
The logical device number of the disc which contains the file label.

address (Double by value, required)
The double-word sector address of the file label on the specified disc.

function (Integer by value, required)
0 to read the file label, 1 to write the label.

label (Array, required)
A 128 word array for the file label.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode. Any errors on the logical device number or disc address may crash the system. Split-Stack calls not permitted.

Copyright (C) 1989 Propellerhead Publishing

DISCUSSION

When reading, the label is checked to insure it is valid. On writing, the checksum is automatically updated.

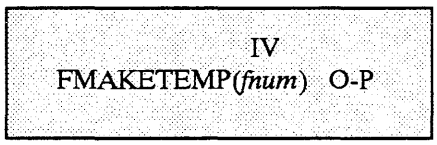
Copyright (C) 1989 Propellerhead Publishing

FMAKETEMP

PURPOSE

Converts an opened permanent file into an opened temporary file.

SYNTAX



PARAMETERS

fnum (Integer by value, required)
The number of the opened permanent file to be changed to a temporary file.

CONDITION CODES

CCL	Error occurred: invalid fnum, wrong file type, file ldev not disc, not exclusive access, label error, file not permanent, duplicate temporary file or job directory table overflow.
CCE	Successful.
CCG	Not returned.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

None.

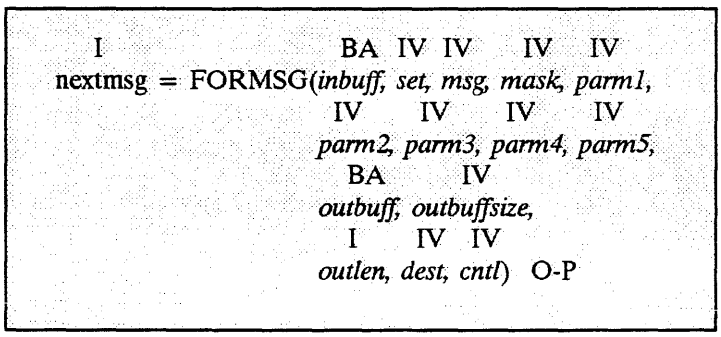
Copyright (C) 1989 Propellerhead Publishing

FORMSG

PURPOSE

Routes a message from the program or the system message catalog to a device, or to an internal buffer.

SYNTAX


I BA IV IV IV IV
nextmsg = FORMSG(inbuff, set, msg, mask, parm1,
IV IV IV IV
parm2, parm3, parm4, parm5,
BA IV
outbuff, outbuffsize,
I IV IV
outlen, dest, cntl) O-P

FUNCTIONAL RETURNS

Returns the next message value in the current message. If none, then -1 returned.

PARAMETERS

inbuff (Byte Array, required)
Buffer containing the message to be formatted. Can contain exclamation marks (!) which FORMSG will substitute with parameters. Ignored if *set* contains a -1. Message must be terminated by a binary zero.

set (Integer by value, required)
This is a flag indicating the source for the message or the message set number itself. The flags are:

Value	Source
-1	The message is in the byte array pointed to by inbuff.
> 0	A message set in the system error message catalog.

When the value is greater than zero, it indicates one of the following message sets in the system error message catalog:

Set Number	Set Name
1	System Messages
2	Command Interp. Messages
3	Misc. Abort Messages

Copyright (C) 1989 Propellerhead Publishing

Set Number	Set Name
4	Program Abort Messages
5	Intrinsic Abort Messages
6	Run-Time Abort Messages
7	CI Warnings
8	File System Errors
9	Loader Errors
10	Create Errors
11	Activate Errors
12	Suspend Errors
13	MYCOMMAND Errors
14	LOCKGLORIN Errors
15	Private Volume Errors
17	Help Messages
18	Graphic Device Messages
19	Serial Disc Errors
20	User Logging Errors
21	Associate Table Errors
22	Laser Printer Errors
25	EPOC User Errors
27	System Internal Errors
28	CIPER Devices Errors

msg (Integer by value, required)
Depending on the contents of 'set', this parameter contains the following:

Value Of set	Contents of Msg
< 0	Ignored
> 0	The number of the message in the specified set.

mask (Logical by value, required)
See the GENMESSAGE intrinsic in the System Intrinsics Manual.

parm1 - parm5 (Logical by value, required)
See the GENMESSAGE intrinsic in the System Intrinsics Manual.

outbuff (Byte Array, required)
Array into which the final message is assembled. If parameters are being substituted, this will contain the final form of the message.

outbuffsize (Integer by value, required)
Size of *outbuff*, in positive bytes.

outlen (Integer, required)
Length of formatted message in *outbuff* returned by FORMSG.

dest (Integer by value, optional)
The destination for the message:

Copyright (C) 1989 Propellerhead Publishing

Value	Destination
< -2	Message written to an open file whose file number is specified as -dest.
= -2	$STDLIST
= -1	No I/O, message returned formatted in the buffer.
= 0	Not used. (Note: if console output is desired, LDEV of console must be derived and passed)
> 0	Message is set to the logical device number specified in dest.

cntl (Logical by value, optional)
If 'dest' greater than zero, this parameter contains the output control used to write to the destination device:

Bits	Description
0:1	Write with cctl %320.
2:2	Funny terminal, translate to APL.
14:2	I/O type - 0 - Standard, wait till all input is done. 1 - Soft Preempt, wait for input unless no characters have been typed, then print. 2 - Hard Preempt, print immediately.

CONDITION CODES

CCL ATTACHIO, PRINT, FWRITE Failure.
CCE Successful.
CCG Something wrong with the parameters, but something printed or returned.

SPECIAL CONSIDERATIONS

Requires privileged mode.

Copyright (C) 1989 Propellerhead Publishing

DISCUSSION

The procedures GENMSG and FORMSG are similar, but the parms to FORMSG are somewhat more straightforward. Also, FORMSG allows the completed, formatted string to be returned to the caller, which GENMSG does not.

Copyright (C) 1989 Propellerhead Publishing

FORMUSERID

PURPOSE

Forms the user ID for the calling user.

SYNTAX

PARAMETERS

user'id (Byte array, required)
Returns the calling user's ID in the form of J/S number, user.account,logon group.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

FREEZESEG'

PURPOSE

Freezes the specified DST in memory and prevents memory manager from relocating it in memory.

SYNTAX

```
                    LV  LV
FREEZESEG'(seg'id, flag)   O-P (MPE-IV)

                    DV  LV
FREEZESEG'(seg'id, flag)   O-P (MPE-V)
```

PARAMETERS

seg'id (Logical by value, required)
The number of the segment to be frozen in memory composed as follows:

(MPE-IV)

Bits	Description
0:1=1	Segment is part of program 1:7 - CSTXBLK INDEX, 8:8 - Segment number(1-255)
0:2=0	Segment is a DST, 2:14 - DST number.
0:2=1	Segment is a SL Segment, 2:14 - SL Segment Number.

(MPE-V)

Word	Bits	Description
0	0:3	Type: 0 = Data Segment 1 = SL Segment 2 = Program Segment 3 = Cache Domain
	4:12	Program CSTXBLK index.
1	0:16	DST, CST, CSTX or CDT Number

flag (Logical by value, required)
This flag is required by the procedure but not used at this entry point.

Copyright (C) 1989 Propellerhead Publishing

FREEZESEG' (Continued)

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

Compliments the procedure UNFREEZESEG'. This is an extra entry point into the LOCKSEG' procedure. This procedure increments the freeze counter in the segment's region header. Until the freeze count falls to zero, Memory Manager will not move the specified segment in memory.

Copyright (C) 1989 Propellerhead Publishing

FREPLY

PURPOSE

Writes a message on $STDLIST and returns a reply from $STDIN.

SYNTAX

```
                L                    BA  IV
error = FREPLY(msg, len)  OP
```

FUNCTIONAL RETURN

A value of TRUE if the procedure was successful, FALSE otherwise.

PARAMETERS

msg (Byte array, required)
When called, contains the message to beprinted on $STDLIST. Upon return, contains the user's reply, up to 8 characters padded with trailing blanks.

len (Integer by value, required)
The length of the message to besent to $STDLIST.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

FROMCALENDAR

PURPOSE

Converts year, day of year into month and day of month.

SYNTAX

```
                                IV    IV     I     I
FROMCALENDAR(year, day'of'year, month, day)
```

PARAMETERS

year (Integer by value, required)
The year (i.e. 1988).

day'of'year (Integer by value, required)
The day of year.

month (Integer, required)
The month of the year (1 = January, 12=December) returned.

day (Integer, required)
The day of the month returned.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

None.

DISCUSSION

See the procedure TOCALENDAR.

Copyright (C) 1989 Propellerhead Publishing

FSCLOSE

PURPOSE

Close/delete files contained in the Input/Output spool. (This is a secondary entry point into the FCLOSE intrinsic).

SYNTAX

```
                    IV    IV  IV
FSCLOSE(fnum, disp, sec)
```

PARAMETERS

fnum (Integer by value, required)
This is the file number returned by the FSOPEN intrinsic when the output spoolfile was originally opened.

disp (Integer by value, required)
This variable represents the disposition of the file when it is closed. The values for this parameter corresponds to those of the FCLOSE intrinsic.

sec (Integer by value, required)
This parameter indicates the level of security placed on the file once it is closed. The possible values for this parameter also correspond to those of the FCLOSE intrinsic.

CONDITION CODES

See FCLOSE intrinsic.

SPECIAL CONSIDERATIONS

Requires privilege mode.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

FSECTORS

PURPOSE

Determines how many sectors are used by a file.

SYNTAX

```
D                                LA
sectors = FSECTORS(file'label)  O-P
```

FUNCTION RETURN

The size of the file in sectors.

PARAMETERS

dst (Logical array, required)
The file label of the file being examined.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

FSOPEN

PURPOSE

Entry point into FOPEN to access file contained in the input/output spool queues.

SYNTAX

```
 L                BA LV   LV  IV    BA  BA
fnum = FSOPEN(file, fopts, aopts, xdd, device, fmsg,
                 IV    IV     IV    DV  IV  IV
              labels, blkfac, numblk, recs, exts, alloc,
               IV
              code) O-V
```

FUNCTIONAL RETURN

The returned value is the file number used to access the file in subsequent file intrinsics.

PARAMETERS

file (Byte array, optional)
Name of file, in ascii, delimited by blank or non-printable character.

fopts (Logical by value, optional)

```
%305 --     OLD, PERM, SYSTEM DOMAIN,
            ASCII, SPECIAL VARIABLE
```

aopts (Logical by value, optional)

```
%530 --     SQUEEZE EXTENTS, MULTI-RECORD
            READ, EXCLUSIVE, NOBUF

%520 --     (same except NOSQUEEZE)

NOTE --     squeeze flag located in
            outflags.(7:1) from
            devrecflags.
```

xdd (Integer by value, optional)
The xdd table and the address within that table of the entry to be opened formatted as follows:

Copyright (C) 1989 Propellerhead Publishing

Bits	Contents
0:1	0 = Input Device Directory Entry 1 = Output Device Directory Entry
1:15	Entry Address In Directory Table

device (Byte array, optional)
Unknown.

fmsg (Byte array, optional)
Unknown.

labels (Integer by value, optional)
Unknown.

blkfac (Integer by value, optional)
Unknown.

numblk (Integer by value, optional)
Unknown.

recs (Double by value, optional)
Unknown.

exts (Integer by value, optional)
Unknown.

alloc (Integer by value, optional)
Unknown.

code (Integer by value, optional)
Unknown.

CONDITION CODES

See the FOPEN intrinsic.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

Once the spool file has been opened, all the normal file intrinsics can be applied. The same is true for obtaining error code should this procedure fail. Note, see the procedure FSCLOSE for closing spool files opened using this method.

Specifying an invalid XDD entry address may cause a system failure!

Copyright (C) 1989 Propellerhead Publishing

GENMSG

PURPOSE

Prints an error message on the requested device.

SYNTAX

```
  I                  IV IV    LV  LV    LV    LV
nextmsg = GENMSG(set, msg, mask, parm1, parm2, parm3,
                      LV    LV    IV  LV    LV
                  parm4, parm5, dest, reply, offset,
                        IV    LV
                  dataseg, cntl) O-V, O-P
```

FUNCTIONAL RETURNS

Returns the next message value in the current message. If none, then -1 returned.

PARAMETERS

set (Integer by value, required)
This is a flag indicating the source for the message or the message set number itself. The flags are:

Value	Source
-1	The message is in the byte array pointed to by 'msg'.
-2	Same as -1 except the write has cctl %320.
> 0	A message set in the system error message catalog.

When the value is greater than zero, it indicates one of the following message sets in the system error message catalog:

Set Number	Set Name
1	System Messages
2	Command Interp. Messages
3	Misc. Abort Messages
4	Program Abort Messages
5	Intrinsic Abort Messages
6	Run-Time Abort Messages
7	CI Warnings
8	File System Errors
9	Loader Errors

Copyright (C) 1989 Propellerhead Publishing

s

Set Number	Set Name
10	Create Errors
11	Activate Error
12	Suspend Errors
13	MYCOMMAND Errors
14	LOCKGLORIN Errors
15	Private Volume Errors
17	Help Messages
18	Graphic Device Messages
19	Serial Disc Errors
20	User Logging Errors
21	Associate Table Errors
22	Laser Printer Errors
25	EPOC User Errors
27	System Internal Errors
28	CIPER Devices Errors

msg (Integer by value, required)
Depending on the contents of 'set', this parameter contains the following:

Value Of set	Contents of Msg
< 0	Pointer to a byte array which contains the message. (Byte array MUST terminate with %0)
> 0	The number of the message in the specified set.

mask (Logical by value, optional)
See the GENMESSAGE intrinsic in the System Intrinsics Manual.

parm1 - parm5 (Logical by value, optional)
See the GENMESSAGE intrinsic in the System Intrinsics Manual.

dest (Integer by value, optional)
The destination for the message:

Value	Destination
< -2	Message written to an open file whose file number is specified as -dest.
= -2	$STDLIST
= -1	No I/O, message returned formatted in the buffer.
= 0	Console, soft-preempted.
> 0	Message is set to the logical device number specified in dest.

reply (Logical by value, optional)
Format of requested reply if destination is the console. If zero, no reply requested.

Copyright (C) 1989 Propellerhead Publishing

Bits	Description
0:8	Maximum length of reply (for type 2).
13:2	Type of reply 0 -- Number, 1 -- Yes/No, 2 -- String

offset (Logical by value, optional)
Word address to which the reply is to be sent. If 'dataseg' specified, data segment relative, otherwise DB relative.

Reply Type	Description
0 (Num)	Word 0 contains the number.
1 (Y/N)	Word 0 TRUE for 'yes', FALSE for 'no'.
2 (str)	Word 0 length of string, String starts in word 1.

dataseg (Integer by value, optional)
If reply requested, indicates the data segment for the reply. If omitted, reply is returned to stack.

cntl (Logical by value, optional)
If 'dest' greater than zero, this parameter contains the output control used to write to the destination device:

Bits	Description
0:1	Write with cctl %320.
2:2	Funny terminal, translate to APL.
14:2	I/O type - 0 - Standard, wait till all input is done. 1 - Soft Preempt, wait for input unless no characters have been typed, then print. 2 - Hard Preempt, print immediately.

CONDITION CODES

CCL ATTACHIO, PRINT, FWRITE Failure.
CCE Successful.
CCG Something wrong with the parameters.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

Copyright (C) 1989 Propellerhead Publishing

The procedure FORMSG will return the formatted message to the caller.

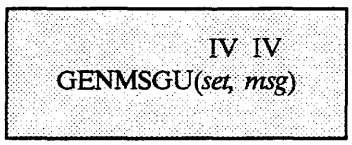
Copyright (C) 1989 Propellerhead Publishing

GENMSGU

PURPOSE

Prints the system error message on the user's terminal.

SYNTAX



PARAMETERS

set (Integer by value, required)
The number of the message set containing the error message to be printed. The system message sets are as follows:

Set Number	Set Name
1	System Messages
2	Command Interp. Messages
3	Misc. Abort Messages
4	Program Abort Messages
5	Intrinsic Abort Messages
6	Run-Time Abort Messages
7	CI Warnings
8	File System Errors
9	Loader Errors
10	Create Errors
11	Activate Errors
12	Suspend Errors
13	MYCOMMAND Errors
14	LOCKGLORIN Errors
15	Private Volume Errors
17	Help Messages
18	Graphic Device Messages
19	Serial Disc Errors
20	User Logging Errors
21	Associate Table Errors
22	Laser Printer Errors
25	EPOC User Errors
27	System Internal Errors
28	CIPER Devices Errors

msg (Integer by value, required)
The number of the message to be printed.

Copyright (C) 1989 Propellerhead Publishing

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

None.

DISCUSSION

Unlike the intrinsic GENMESSAGE, the system message file CATALOG.PUB.SYS does not have to be opened before calling this procedure.

Copyright (C) 1989 Propellerhead Publishing

GET'DSDEVICE

PURPOSE

Returns information on potential DS devices.

SYNTAX

```
                I                          IV
info = GET'DSDEVICE(ldev)  O-P
```

FUNCTIONAL RETURN

The possible values for *info* are:

Value	Meaning
-2	No DS lines are configured.
-1	Invalid LDEV number.
0	Not DS related device.
1	DS device (INP).
2	DS device.
3	DS pseudo terminal.
4	PAD terminal.

PARAMETERS

ldev (Integer by value, required)
The logical device number of the device to examine.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

Copyright (C) 1989 Propellerhead Publishing

None.

Copyright (C) 1989 Propellerhead Publishing

GET'FILECODE

PURPOSE

Returns the file code assigned to the specified file mnemonic.

SYNTAX

```
                                I       BA    I
GET'FILECODE(fcode, mnemonic, len)  O-P
```

PARAMETERS

fcode (Integer, required)
The file code assigned to the specified file mnemonic.

mnemonic (Byte Array, required)
The file code mnemonic to be interpreted.

len (Integer, required)
The length of the mnemonic in characters.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

GET'FILEMNEMONI

PURPOSE

Returns the mnemonic associated with the specified file code.

SYNTAX

```
                                    IV      BA   I
GET'FILEMNEMONI(fcode, mnemonic, len) O-P
```

PARAMETERS

fcode (Integer by value, required)
The file code to be interpreted.

mnemonic (Byte Array, required)
The file code mnemonic returned to the caller.

len (Integer, required)
The length of the mnemonic in characters.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

GET'SYS'PRIV

PURPOSE

Determines if the specified code segment is privileged or system.

SYNTAX

```
                    L                    IV
priv'sys = GET'SYS'PRIV(cst)  O-P
```

FUNCTIONAL RETURN

Returns TRUE if specified code segment is privileged or system, FALSE otherwise.

PARAMETERS

cst (Integer by value, required)
The code segment index of the segment to be tested. May be system (i.e. < %300) or user.

CONDITION CODES

CCL CST is not a valid number.
CCE CST identified and response provided.
CCG CST is unassigned.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

GETCLASS

PURPOSE

Obtains information on devices in a particular class.

SYNTAX

```
L                          IA  LV  IV   IV   IA
flag = GETCLASS(buffer, all, addr, indx, name)  O-P, O-V
```

FUNCTIONAL RETURN

A flag indicating success (TRUE) or failure (FALSE).

PARAMETERS

buffer (Integer array, required)
This array contains the information on the requested device class. The array should be at least 5 words in length and will contain the following information upon return:

Word	Contents
0	Pointer to entry in the device class table.
1	Class index (not returned if 'claddr' specified).
2	Word 4 of class table entry.
3	Number of devices, first logical device number.
4	First LDEV in class.
5+	Remainder of devices within the class (only returned if 'all' = TRUE).

all (Logical by value, required)
A flag indicating to return all device numbers (TRUE) or only words 0-4 in the buffer (FALSE).

addr (Integer by value, optional)
Pointer to the class entry in the device class table. If this is specified, do not specify the *indx* or *name* parameters.

Copyright (C) 1989 Propellerhead Publishing

GETCLASS(continued)

indx (Integer by value, optional)
Class index. If this is specified, do not specify the *addr* or *name* parameters.

name (Integer array, optional)
An array containing the name of the device class. If this is specified, do not specify the *addr* or *indx* parameters.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

The LDTSIR must be locked prior to the procedure call if updating the table.

Potential system failures include:

```
360 - Device class table index not within table limits.
365 - More than one selection parameter was specified.
367 - Buffer or all flag not specified.
```

Copyright (C) 1989 Propellerhead Publishing

GETCLOCK

PURPOSE

Initiate the shared clock interface and provide a user procedure that is performed when the interrupt occurs.

SYNTAX

```
                I      IV    I     IV    IV
GETCLOCK(interval, plabel, error, dst1, dst2)  O-P, O-V
```

PARAMETERS

interval (Integer, required)
The requested interrupt rate, in milliseconds. A value of 0 will cause everything to be setup but not start calling the user's procedure. Then the next call with a 0 plabel and a non-zero interval will start the data collection.

plabel (Integer by value, required)
The label of the user's data collection procedure. If a value of 0 is specified, the clock is reset but control of the clock is not released.

error (Integer, required)
A returned value indicating the error encountered:

```
1 - Missing required parameter.
2 - Clock interface already in use.
3 - Invalid interrupt rate.
4 - Invalid plabel.
5 - Cannot freeze code segment
    containing the user's data
    collection procedure.
6 - Cannot freeze data segment.
7 - Data collection procedure
    calling non-resident segment.
8 - Unable to obtain CST entry.
```

dst1 (Integer by value, optional)
The DST number of a data segment to be used by the data collection procedure.

dst2 (Integer by value, optional)
The DST number of a second data segment to be used by the data collection procedure.

Copyright (C) 1989 Propellerhead Publishing

GETCLOCK(continued)

CONDITION CODES

CCL Failure, see error code returned in error parameter.
CCE Successful.
CCG Not returned.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

GETDATASEG + GetdataSegC

PURPOSE

Acquires a data segment and returns its DST number.

SYNTAX

I

```
                IV  IV
dst = GETDATASEG(len, vmem)   O-P
```

FUNCTIONAL RETURN

The GETDATASEG procedure returns an integer value equal to the DST number of the acquired data segment.

PARAMETERS

len (Integer by value, required)
The desired length of the data segment to be obtained. This value must be rounded upwards to the nearest multiple of 4 words.

vmem (Integer by value, required)
The amount of virtual memory to be allocated for the data segment. This value must be a rounded upwards to a multiple of 512 words and must be large enough to accommodate the specified length of the data segment.

CONDITION CODES

CCL Not enough virtual memory available.
CCE Successful.
CCG No DST table entries available.

SPECIAL CONSIDERATIONS

Requires privileged mode.

Copyright (C) 1989 Propellerhead Publishing

GETDATASEG(continued)

DISCUSSION

Data segments obtained using this procedure are not reflected in the calling process's stack or job data segment directory portion of the JDT table. Therefore, the data segment will remain in the system when the creating process terminates or logs off. It becomes the responsibility of the creator to manage the DST number and delete the data segment when its use is no longer necessary. The procedure used to release the data segment is RELDATASEG.

Possible system failures include:

```
680 - ATTACHIO unable to zero the data segment.
```

Copyright (C) 1989 Propellerhead Publishing

GETDEVINFO

PURPOSE

Obtains the LDT, LPDT and DCT table entries for a device.

SYNTAX

```
                                I                    BA      IA
error = GETDEVINFO(device, deviceinfo)  O-P
```

FUNCTIONAL RETURN

An error code is returned with the following meaning:

Error	Meaning
0	Successful
1	Invalid device class
2	Unknown device class
3	Unknown device number
-1	Virtual device

PARAMETERS

device (Byte array, required)
An ASCII array containing the device class, DS node or a logical device number. A device class name must be 8 characters, left-justified and padded with trailing blanks. A DS node must end with a "#". A logical device number must be passed as ASCII characters.

deviceinfo (Integer array, required)
This array contains the LDT, LPDT and DCT entries for the requested device. Its contents depend on the contents in the 'device' parameter.

Device Class	Data Returned
	Word 0 - DCT Index
	Word 1 - Device Type as in the DCT
	Word 2 - Word 4 of DCT
	Word 3 - Unused.
	Word 4/8 - LDT entry of first device
	Word 9/10 - Unused.

Copyright (C) 1989 Propellerhead Publishing

Device	Data Returned
DS Node	Word 0 - Unused. Word 1 - 41 (DS device type) Word 2/10 - Unused
Ldev	Word 0 - Device number Word 1 - Device Type as it appears in the LDT Word 2/5 - LPDT entry Word 6/10 - LDT entry

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

GETJITDST

PURPOSE

Returns the DST number of the calling user's Job Information Table (JIT).

SYNTAX

```
    I
dst = GETJITDST   O-P
```

FUNCTIONAL RETURN

The GETJITDST procedure returns an integer value representing the number of the extra data segment (DST) containing the calling user's Job Information Table (JIT).

PARAMETERS

None.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

GETJOBN

PURPOSE

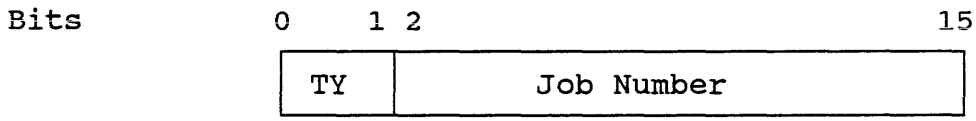
Returns the job number of the calling user.

SYNTAX

```
L
job = GETJOBN  O-P
```

FUNCTIONAL RETURN

The GETJOBN procedure returns an logical value containing the job number of assigned to the calling user's job/session. It is in the form:



Where: TY = 1 - Session
2 - Job

PARAMETERS

None.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

None.

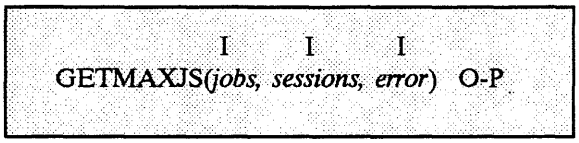
Copyright (C) 1989 Propellerhead Publishing

GETMAXJS

PURPOSE

Returns the maximum number of jobs and sessions specified during system configuration.

SYNTAX



PARAMETERS

jobs (Integer, required)
The maximum number of jobs.

sessions (Integer, required)
The maximum number of sessions.

error (Integer, required)
An error value returned. If zero, call was successful. If greater than zero, the value represents the file system error that occurred while attempting to access the file CONFDATA.PUB.SYS.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

GETPCBXMEASINFO

PURPOSE

Retrieve information for a PIN from its PCB entry and the PCBX fixed area in the stack.

SYNTAX

```
                                    I   LA
GETPCBXMEASINFO(pin, info)  O-P
```

PARAMETERS

pin (Integer, required)
The PIN number of the desired process.

info (Logical array, required)
An array in which the information is returned:

Word	Information
0	(0:2) = 1 - Session 2 - Job
	(2:14) = Job Number.
1	(0:1) = Linear Queue
	(1:1) = "C" Queue
	(2:1) = "D" Queue
	(3:1) = "E" Queue
	(4:12) = Stop type from PBX Fixed %101.
2-3	Stop time from PBX fixed word %102-%103.

CONDITION CODES

CCL Not a valid process.
CCE Successful.
CCG Not returned.

SPECIAL CONSIDERATIONS

Requires privileged mode.

Copyright (C) 1989 Propellerhead Publishing

DISCUSSION

Does not return information on system processes.

Copyright (C) 1989 Propellerhead Publishing

GETSALT

PURPOSE

Calculates the password encryption seed based upon the user, group and/or account names.

SYNTAX

```
 I                    BA    BA    BA
error = GETSALT(name, account, seed)
```

FUNCTIONAL RETURN

A zero (0) indicates successful, non-zero indicates an error occurred.

PARAMETERS

name (Byte array, required)
An array containing the user, group or account name depending on the which level password is to be encrypted. Must be left justified, upshifted and padded with trailing blanks.

account (Byte array, required);
An array continuing the account name or the constant "SYSTEM" if an account level password is being encrypted. Must also be left justified, upshifted and padded with trailing blanks.

seed (Byte array, required)
A seed which is used by the CRYPT procedure to encrypt the password.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode. Also, this is a PASCAL procedure and only callable from SPL or PASCAL due to the parameter checking level.

Copyright (C) 1989 Propellerhead Publishing

DISCUSSION

To determine if a password is encrypted, the following flags exist in the appropriate directory entries:

Entry Type	Word	Flag
Account%33	2:1	Password Encrypted.
	3:1	Password Required.
Group	%251:1	Password Encrypted.
User	%212:1	Password Encrypted.
	3:1	Password Required.
	4:1	Password Expired.

Copyright (C) 1989 Propellerhead Publishing

GETSECINFO

PURPOSE

Obtains security configuration information from the system global security DST.

SYNTAX

```
 L                              LA  IV    IV  IV
flag = GETSECINFO(buffer, table, offset, len)  O-P
```

FUNCTIONAL RETURN

A TRUE value indicates information successfully returned. A FALSE value indicates an error occurred.

PARAMETERS

buffer (Logical array, required)
An array to hold the returned information.

table (Integer by value, required);
Identifies the table number within the DST to be accessed.

offset (Integer by value, required)
An offset into the requested table.

len (Integer by value, required)
The number of words to be returned.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

GETSIR

PURPOSE

Obtains a lock on the specified system internal resource (SIR) number prior to modifying the associated data. If the SIR is currently locked by another process, the calling process will be impeded until the SIR becomes available. This prevents concurrent modification of data.

SYNTAX

```
        L                  LV
sir'flag = GETSIR(sir) O-P
```

FUNCTION RETURN

The logical value returned by GETSIR is required by the RELSIR procedure to reverse the process and unlock the SIR. The values are:

Value	Meaning
0	The calling process did not own any SIRs before the call.
1	The calling process already owned at least 1 SIR but not the one requested.
3	The calling process already owned this SIR.

PARAMETERS

sir (Logical by value, required)
The system internal resource number associated with the resource to be locked. The SIR number are assigned as follows:

SIR Number	Resource Name
1	(% 1) Load Process SIR
2	(% 2) Lock Segment SIR
3	(% 3) Input Device Directory (IDD)
4	(% 4) Output Device Directory (ODD)
5	(% 5) Process Tree Structure
6	(% 6) Scheduling Queue
7	(% 7) CST Entries
8	(%10) System Directory
9	(%11) Logical Phys. Device Table (LPDT)
10	(%12) Logical Device Table (LDT)
11	(%13) Storage in Overlay Area
12	(%14) Disc Free Space Table

Copyright (C) 1989 Propellerhead Publishing

13	(%15) Job Process Count Table (JPCNT)
14	(%16) Job Cutoff Table (JCUT)

SIR Number	Resource Name
15	(%17) Job Master Table (JMAT)
16	(%20) File Multi-Access Vector Table (FMAVT)
17	(%21) Loader Segment Table
18	(%22) VDD
19	(%23) Spool
20	(%24) Message Catalogue
21	(%25) Reply Information Table (RIT)
22	(%26) Volume Table
23	(%27) Welcome Message SIR
24	(%30) Association Table
25	(%31) CS Allocate SIR
26	(%32) Logging Buffer
27	(%33) Mounted Volume Table (MVTAB)
28	(%34) MEASSIR
29	(%35) Private Volume User Table
30	(%36) IMAGE
31	(%37) KSAM
32	(%40) User Logging
33	(%41) Debug Breakpoint Table
34	(%42) Process Control Block (PCB)
35	(%43) Sub-Queue Mapping Table
36	(%44) CILOG
37	(%45) File Integrity
38	(%46) RIN
39	(%47) Tape Labels
40	(%50) Device Class Table
41	(%51) Cold Load Table
42	(%52) 1st Job
43	(%53) 2nd Job
.	.
.	.

CONDITION CODES

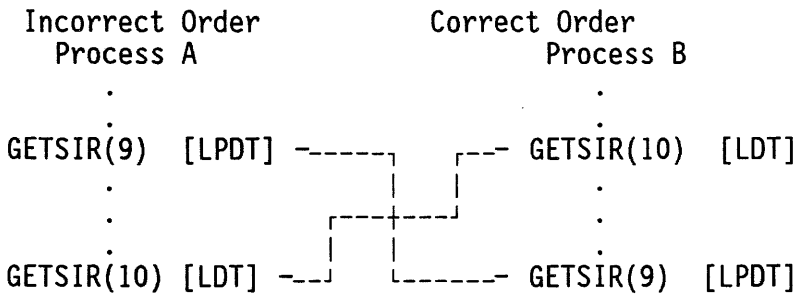
Unchanged.

Copyright (C) 1989 Propellerhead Publishing

GETSIR (continued)

SPECIAL CONSIDERATIONS

Requires privileged mode. Also, multiple SIR allocation requires care to avoid process deadlock situations. The rule to follow is to never attempt a GETSIR of lower rank than the SIR currently held (if any). For example:



Refer to Chapter 5 of the V-MIT MPE-V Tables Manual for a full listing of the precedence rules for all SIRs. Multiple SIRs must be released in a LIFO fashion. Also, never terminate with a locked SIR. This may result in a system failure 314. The calling process should either be placed in CRITICAL mode (see SETCRITICAL) or the BREAK key should be disable while SIRs are locked to prevent user aborts. Locked SIRs should be released as quickly as possible.

DISCUSSION

This procedure and it's complement procedure RELSIR are used to lock a system internal resource number (SIR) which is associated with an internal resouce to prevent concurrent access.

Copyright (C) 1989 Propellerhead Publishing

GETSTATISTICS

PURPOSE

Returns information from the MPE Internal Measurement Facility.

SYNTAX

```
 I                                IV    IV       IV
error = GETSTATISTICS(class, subclass, startingitem,
                                  IV      IA
                      wordcount, where) O-P
```

FUNCTIONAL RETURN

An error code indicating the error encountered while attempting to obtain the specified information as follows:

Value	Meaning
0	Successful.
1	MPE Internal Measurement Facility not enabled.
2	Target array out of bounds.
3	Word count out of range.
4	Bad starting item parameter.
5	Bad subclass parameter.
6	Bad class parameter.

PARAMETERS

class (Integer by value, required)
An value specify the statistical class desired. The classes are identified as follows:

Class	Description
0	System global information, disc, printer and tape usage.
14	I/O statistics.
15	Individual process data.

subclass (Integer by value, required)
A value specifying which portion of the statistics within the class:

Class	Subclass
0	Global Information, 100 words.
1	Disc Access Data, 20 words per disc.

Copyright (C) 1989 Propellerhead Publishing

```
    2   Printer Information, 4 words per
        printer.
Class       Subclass
    3   Tape Drive Data 4 words per tape
        unit.
   14   Builds circular queue of IOQ/DRQ
        entries in extra data segment.
   15   Subclass not required, set to zero.
```

startingitem (Integer by value, required)
The starting word within the specific class & subclass which is to be returned.

wordcount (Integer by value, required)
The number of words to return.

where (Integer array, required)
An array in which the requested data is returned.

CONDITION CODES

CCL Failure.
CCE Successful.
CCG Not returned.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

The entry point FGETSTATISTICS can be used to bypass the parameter checking performed by the primary entry point.

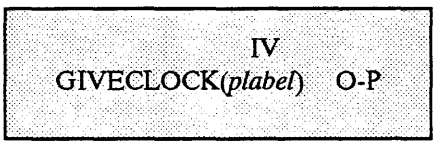
Copyright (C) 1989 Propellerhead Publishing

GIVECLOCK

PURPOSE

Releases the shared clock interface.

SYNTAX



PARAMETERS

plabel (Integer by value, required)
The label of the user's data collection procedure. If a value of 0 is specified, the clock is reset but control of the clock is not released.

CONDITION CODES

CCL Failure, unable to unlock/unfreeze either the code or data segment(s).
CCE Successful.
CCG Failure, user does not own the clock.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

HELPPROC

PURPOSE

This procedure is the HELP subsystem driver.

SYNTAX

```
                    IV  IV  BA  BA  I     LV
HELPPROC(fnum, out, cmd, base, error, interact)
```

PARAMETERS

fnum (Integer by value, required)
This is the file number of the opened HELP catalog file. The access options (AOPTS) should be set to %300 when the file is opened.

out (Integer by value, required)
The file number to which the HELP output is to be sent. To send the output to $STDLIST, use 2.

cmd (Byte Array, required)
This byte array contains the key or HELP command to be executed, terminated by a carriage return (%15).

base (Byte Array, required)
(Unknown, use a carriage return (%15).)

error (Integer, required)
The return error code.

interact (Logical by value, required)
A flag indicating whether to stop after the first error (FALSE) or to continue (TRUE).

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

IDENTIFY'TERMIN

PURPOSE

Identify the model number of the HP terminal currently being used by the calling process.

SYNTAX

```
                                  LV    BA   I     L        I
IDENTIFY'TERMIN(term, hp'term, type, term'type, line'plex,
                                  BA        BA       I
                pri'status, sec'status, ret'status)
```

PARAMETERS

term (Logical by value, required)
A word identifier supplying the file number of the terminal.

hp'term (Byte array, required)
An array into which the model number of the terminal device is returned. This array should be at least 6 bytes long. Possible values include HP2621, HP2624, HP2626, HP2645, etc.

type (Integer, required)
An integer value used to identify the terminal. A table showing the value associated with each terminal model is located in the VPLUS documentation. Found under the description for the COMAREA item *IDENTIFIER* (word 58/59).

term'type (Logical, required)
The current terminal type value returned to the calling process.

line'plex (Integer, required)
Code indicating half or full-duplex line.

pri'status (Byte array, required)
A byte array into which is returned the primary status bits obtained by the procedure from the terminal. Must be large enough to contain 8 bytes of status information.

sec'status (Byte array, required)
A byte array into which is returned the secondary status bits (if any) obtained by the procedure. Must be large enough to contain 8 bytes of status information.

ret'status (Integer, required)
A word identifier returned to the calling process to indicate the success of the procedure. A zero value indicates the procedure was successful in determining the terminal model number. A positive value indicates the file system error code encountered by the procedure. A negative value also indicates a failure, however, it's significance is not known to the author.

Copyright (C) 1989 Propellerhead Publishing

IDENTIFY'TERMIN(continued)

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Prior to calling the IDENTIFY'TERMIN procedure, the terminal device must be successfully opened and the terminal type must be check to insure it is type 10. This can be accomplished as follows:

```
FCONTROL(terminal, 39, term'type);
IF term'type <> 10 THEN
   BEGIN
        term'type := 10;
        FCONTROL(terminal, 38, term'type);
   END;
```

DISCUSSION

None.

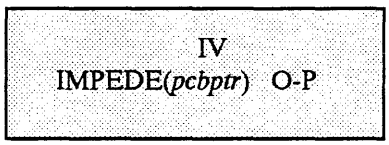
Copyright (C) 1989 Propellerhead Publishing

IMPEDE

PURPOSE

Impedes the process located at the specified address in the PCB table.

SYNTAX



PARAMETERS

pcbptr (Integer by value, required)
The PCB relative base address of the process to be impeded. If zero is specified, the current process is impeded.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode. Split-stack calls permitted.

DISCUSSION

There is no mechanism for keeping track of the impeded process. It is the responsibility of the caller to insure the impeded process is awaken.

Compliments the procedure UNIMPEDE.

Copyright (C) 1989 Propellerhead Publishing

IOCONTROL

PURPOSE

Procedure to perform I/O control operations on the specified device.

SYNTAX

```
 L                              IV     IV
success = IOCONTROL(device, function)  O-P
```

FUNCTIONAL RETURN

This procedure returns a TRUE/FALSE indicating whether or not the operation was successful.

PARAMETERS

device (Integer by value, required)
The number of the logical device on which the function is to be performed.

function (Integer by value, required)
This parameter indicates the type of operation to be performed on the specified device. These codes are device-dependent and are the same as those used by ATTACHIO.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

The calling process is blocked until the requested I/O function has been completed.

Copyright (C) 1989 Propellerhead Publishing

IOFREEZE

PURPOSE

If the specified segment memory is in memory, it is locked to prevent swaps to virtual.

SYNTAX

```
                    LV  LV  LV
IOFREEZE(seg, flag, pin)   O-P (MPE-IV)

                    DV  LV  LV
IOFREEZE(seg, flag, pin)   O-P (MPE-V)
```

PARAMETERS

seg (Logical by value, required)
A word identifying the segment to be locked in memory:

(MPE-IV)

Bits	Description
0:1=1	Segment is part of program 1:7 - CSTXBLK INDEX, 8:8 - Segment number(1-255)
0:2=0	Segment is a DST, 2:14 - DST number.
0:2=1	Segment is a SL Segment, 2:14 - SL Segment Number.

(MPE-V)

Word	Bits	Description
0	0:3	Type: 0 = Data Segment 1 = SL Segment 2 = Program Seg 3 = Cache Domain
	4:12	Program CSTXBLK index.
1	0:16	DST, CST, CSTX or CDT Number

flag (Logical by value, required)
This flag indicates the type of segment to be locked:

TRUE - The segment identifier 'seg' contains a DST number.

Copyright (C) 1989 Propellerhead Publishing

```
FALSE -The segment identifier 'seg' is either a CST
       number if PIN = zero or a CST block index
       relative to %300 if PIN not equal to zero.
```

pin (Logical by value, required)
If locking a CST block index, this contains the PIN number of the process. Otherwise zero.

CONDITION CODES

CCL The specified segment was not in memory.
CCE Successful.
CCG Not returned.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

Performs the same function as FREEZE, but signals the Memory Manager that the segment will be frozen for a short time only, until an I/O operation completes. Compliments the procedure IOUNFREEZE. Unlike LOCKSEG, this procedure does not make an absent segment present. If the segment is in virtual, the procedure returns a condition code of CCL.

Copyright (C) 1989 Propellerhead Publishing

IOFREEZE'

PURPOSE

If the specified segment memory is in memory, it is locked to prevent swaps to virtual.

SYNTAX

```
                    LV  LV
IOFREEZE'(seg'id, flag)  O-P (MPE-IV)

                    DV  LV
IOFREEZE'(seg'id, flag)  O-P (MPE-V)
```

PARAMETERS

seg'id (Logical by value, required)
The number of the segment to be locked in memory composed as follows:

	Bits	Description
(MPE-IV)	0:1=1	Segment is part of program 1:7 - CSTXBLK INDEX, 8:8 - Segment number(1-255)
	0:2=0	Segment is a DST, 2:14 - DST number.
	0:2=1	Segment is a SL Segment, 2:14 - SL Segment Number.

	Word	Bits	Description
(MPE-V)	0	0:3	Type: 0 = Data Segment 1 = SL Segment 2 = Program Seg 3 = Cache Domain
		4:12	Program CSTXBLK index.
	1	0:16	DST, CST, CSTX or CDT Number

flag (Logical by value, required)
This flag indicates whether the segment will be frozen in memory or not.

TRUE - The segment will be frozen. IOFREEZE' returns only after the segment has migrated to a memory boundary.

Copyright (C) 1989 Propellerhead Publishing

```
FALSE -The segment is not going to be frozen.
       IOFREEZE' returns as soon as the segment
       becomes present. The segment may be moved
       about in memory as needed.
```

CONDITION CODES

CCL The specified segment was not in memory.
CCE Successful.
CCG Not returned.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

Compliments the procedure IOUNFREEZE'.

Copyright (C) 1989 Propellerhead Publishing

IOSTAT

PURPOSE

Returns the status and transmission log/control returns of the request identified by ioqx.

SYNTAX

```
                D                    IV
d'status = IOSTAT(ioqx)  O-P
```

FUNCTION RETURN

The status and transmission log/control return from the IOQ entry indicated by ioqx. This is the same data that would be returned by ATTACHIO if the request were a blocked reqeust. The return has the following format:

Word	Bits	Description
0	0:7	PIN number.
	8:5	Qualifying status.
	13:3	General status: 0 - Pending 1 - Successful 2 - End of file 3 - Unusual condition 4 - Irrecoverable error
1		Number of words (positive) or bytes (negative) transferred.

PARAMETERS

ioqx (Integer by value, required)
The entry number in the IOQ table that IOSTATUS will examine. This number is returned by ATTACHIO if the *flags* parameter indicates a non-blocked request.

Copyright (C) 1989 Propellerhead Publishing

CONDITION CODES

CCL Not returned.
CCE Successful.
CCG I/O not complete.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

See the Expansions section (Section III) for further detail on the status returns.

Copyright (C) 1989 Propellerhead Publishing

IOUNFREEZE

PURPOSE

Unlocks the specified segment to allow it to be swapped to virtual storage.

SYNTAX

```
                                LV  LV LV
IOUNFREEZE(seg, flag, pin)   O-P (MPE-IV)

                                DV  LV LV
IOUNFREEZE(seg, flag, pin)   O-P (MPE-V)
```

PARAMETERS

seg (Logical by value, required)
A word identifying the segment to be unlocked from memory:

(MPE-IV)

Bits	Description
0:1=1	Segment is part of program
1:7	- CSTXBLK INDEX,
8:8	- Segment number(1-255)
0:2=0	Segment is a DST,
2:14	- DST number.
0:2=1	Segment is a SL Segment,
2:14	- SL Segment Number.

(MPE-V)

Word	Bits	Description
0	0:3	Type: 0 = Data Segment 1 = SL Segment 2 = Program Seg 3 = Cache Domain
	4:12	Program CSTXBLK index.
1	0:16	DST, CST, CSTX or CDT Number

flag (Logical by value, required)
This flag indicates the type of segment to be unlocked:

TRUE - The segment identifier 'seg' contains a DST number.

Copyright (C) 1989 Propellerhead Publishing

```
FALSE -The segment identifier 'seg' is either a CST
       number if PIN = zero or a CST block index
       relative to %300 if PIN not equal to zero.
```

pin (Logical by value, required)
If locking a CST block index, this contains the PIN number of the process. Otherwise zero.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

Compliments the procedure IOFREEZE.

Copyright (C) 1989 Propellerhead Publishing

IOUNFREEZE'

PURPOSE

Unlocks the specified DST to allow it to be swapped to virtual storage.

SYNTAX

```
                                LV
IOUNFREEZE'(seg'id)  O-P (MPE-IV)

                                DV
IOUNFREEZE'(seg'id)  O-P (MPE-V)
```

PARAMETERS

seg'id (Logical by value, required)
The number of the segment to be unlocked from memory composed as follows:

(MPE-IV)

Bits	Description
0:1=1	Segment is part of program 1:7 - CSTXBLK INDEX, 8:8 - Segment number(1-255)
0:2=0	Segment is a DST, 2:14 - DST number.
0:2=1	Segment is a SL Segment, 2:14 - SL Segment Number.

(MPE-V)

Word	Bits	Description
0	0:3	Type: 0 = Data Segment 1 = SL Segment 2 = Program Seg 3 = Cache Domain
	4:12	Program CSTXBLK index.
1	0:16	DST, CST, CSTX or CDT Number

Copyright (C) 1989 Propellerhead Publishing

IOUNFREEZE' (continued)

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

Compliments the procedure IOFREEZE'.

Copyright (C) 1989 Propellerhead Publishing

ISSYSCACHABLE

PURPOSE

Determines if disc caching is installed.

SYNTAX

```
                L
cache = ISSYSCACHABLE  O-P
```

FUNCTIONAL RETURN

A value of TRUE if caching is installed, FALSE if not.

PARAMETERS

None.

CONDITION CODES

Not returned.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

JULDATE

PURPOSE

Converts a julian date to gregorian.

SYNTAX

```
                                DV I     I  I
JULDATE(julian, day, month, year)
```

PARAMETERS

day (Double by value, required)
Julian date to be converted to gregorian.

day (Integer, required)
The day of the month returned.

month (Integer, required)
The month of the year (1 = January, 12=December) returned.

year (Integer, required)
The year (i.e. 1988) returned.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

None.

DISCUSSION

Converts a julian calendar date back to gregorian. Using julian dates for calculations eliminates the need to deal with leap years. Also see the procedure DATEJUL.

Copyright (C) 1989 Propellerhead Publishing

LDEVTOTYPE

PURPOSE

Returns device type for specified LDEV number.

SYNTAX

```
                    I                    IV
type = LDEVTOTYPE(ldev) O-P
```

FUNCTIONAL RETURN

The device type of the specified LDEV.

PARAMETERS

ldev (Integer by value, required)
The LDEV number of the device to be examined.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

LDEVTOVTAB

PURPOSE

Converts a disc file extent map from logical device number to volume table indexes.

SYNTAX

```
                                    DA   DA   IV   LV
LDEVTOVTAB(ext'vol, ext'ldev, count, local)  O-P
```

PARAMETERS

ext'vol (Double array, required)
The returned double-word extent map containing volume table indexes.

ext'ldev (Double array, required)
The source double-word extent map containing logical device numbers.

count (Integer by value, required)
The number of extent entries in *ext'ldev*.

local (Logical by value, required)
If TRUE, retrieve the volume table index from the volume table.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode. Cannot be called in split-stack.

DISCUSSION

Process is reversed using the VTABTOLDEV procedure. Also, the *ext'vol* and *ext'ldev* can be the same array.

Copyright (C) 1989 Propellerhead Publishing

LOCKJIR

PURPOSE

Resource lock facility used to coordinate access to a process's Job Directory Table (JDT) table.

SYNTAX

```
          L
jir'flag = LOCKJIR   O-P
```

FUNCTION RETURN

The logical value returned by LOCKJIR is required by the UNLOCKJIR procedure to reverse the process and unlock the JIR.

PARAMETERS

None.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

Complements the UNLOCKJIR procedure.

Copyright (C) 1989 Propellerhead Publishing

LOCKSEG

PURPOSE

Locks the specified segment in memory to prevent swaps to virtual.

SYNTAX

```
                    LV  LV LV
LOCKSEG(seg, flag, pin)   O-P (MPE-IV)

                    DV  LV LV
LOCKSEG(seg, flag, pin)   O-P (MPE-V)
```

PARAMETERS

seg (Logical by value, required)
A word identifying the segment to be locked in memory:

(MPE-IV)

Bits	Description
0:1=1	Segment is part of program 1:7 - CSTXBLK INDEX, 8:8 - Segment number(1-255)
0:2=0	Segment is a DST, 2:14 - DST number.
0:2=1	Segment is a SL Segment, 2:14 - SL Segment Number.

(MPE-V)

Word	Bits	Description
0	0:3	Type: 0 = Data Segment 1 = SL Segment 2 = Program Segment 3 = Cache Domain
	4:12	Program CSTXBLK index.
1	0:16	DST, CST, CSTX or CDT Number

flag (Logical by value, required)
This flag indicates the type of segment to be locked:

TRUE - The segment identifier 'seg' contains a DST number.

Copyright (C) 1989 Propellerhead Publishing

```
FALSE -The segment identifier 'seg' is either a CST
       number if PIN = zero or a CST block index
       relative to %300 if PIN not equal to zero.
```

pin (Logical by value, required)
If locking a CST block index, this contains the PIN number of the process. Otherwise zero.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

Compliments the procedure UNLOCKSEG.

OLd - migrate TO lockseg !

Copyright (C) 1989 Propellerhead Publishing

LOCKSEG'

PURPOSE

Locks the specified DST in memory to prevent swaps to virtual.

SYNTAX

```
                              LV  LV
LOCKSEG'(seg'id, flag)   O-P (MPE-IV)

                              DV  LV
LOCKSEG'(seg'id, flag)   O-P (MPE-V)
```

PARAMETERS

seg'id (Logical by value, required)
The number of the segment to be locked in memory composed as follows:

(MPE-IV)

Bits	Description
0:1=1	Segment is part of program 1:7 - CSTXBLK INDEX, 8:8 - Segment number(1-255)
0:2=0	Segment is a DST, 2:14 - DST number.
0:2=1	Segment is a SL Segment, 2:14 - SL Segment Number.

(MPE-V)

Word	Bits	Description
0	0:3	Type: 0 = Data Segment 1 = SL Segment 2 = Program Segment 3 = Cache Domain
	4:12	Program CSTXBLK index.
1	0:16	DST, CST, CSTX or CDT Number

flag (Logical by value, required)
This flag indicates whether the segment will be frozen in memory or not.

TRUE - The segment will be frozen.
LOCKSEG' returns only after the

Copyright (C) 1989 Propellerhead Publishing

segment has migrated to a memory boundary.

FALSE - The segment is not going to be frozen. LOCKSEG' returns as soon as the segment becomes present. The segment may be moved about in memory as needed.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

Compliments the procedure UNLOCKSEG'.

Copyright (C) 1989 Propellerhead Publishing

LOGICALCST

PURPOSE

Returns the number and type associated with the CST contained in the status word.

SYNTAX

```
        D                                IV
cst'type = LOGICALCST(status'seg)
```

FUNCTIONAL RETURN

The LOGICALCST procedure returns a double word value constructed as follows:

Word	Value
0	Type of CST segment: 0 - System SL segment 1 - PUB SL segment 2 - Group SL segment 3 - Program segment
1	The CST number assigned to the segment.

PARAMETERS

status'seg (Integer by value, required)
The CST number contained in the status register.

CONDITION CODES

CCL The specified CST is invalid.
CCE Successful.
CCG Not returned.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

LUN

PURPOSE

Returns the logical device number of a disc given its volume table index.

SYNTAX

```
I              IV      IV
ldev = LUN(vtabx, mvtabx)  O-P
```

FUNCTIONAL RETURN

The logical device number of the disc.

PARAMETERS

vtabx (Integer by value, required)
The volume table index as contained in the system directory.

mvtabx (Integer by value, required)
The mounted private volume table index, zero if only system discs available.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

Many of the double word disc addresses that are returned from directory calls or stored in various system tables actually contain a volume index in the first 8 bits, not a logical device number. This routine obtains the necessary device number that the disc address references.

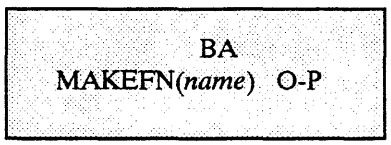
Copyright (C) 1989 Propellerhead Publishing

MAKEFN

PURPOSE

Generates a filename using the current date and time.

SYNTAX



PARAMETERS

name (Byte array, required)
Returns a 9 character filename consisting of the letter "S" followed by the current 3-digit julian date, 2-digit minutes and 2-digit seconds padded with trailing blanks.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

A secondary entry point, MAKEFN', can be used whenever duplicate names occur. In this instance an attempt is made to stay within the same day and hour by incrementing the minutes from the value 60. If this should fail, the hour is reset and incremented beginning at the value 24. Should this fail, the day is reset and all values are incremented beginning with the value 367 for the day, 00 for the hour and 00 for the minute. As a last resort, the day, hour and minutes are reset to zero.

Copyright (C) 1989 Propellerhead Publishing

MASTEROP

PURPOSE

Indicates if the current logon device is the system console.

SYNTAX

```
I
console = MASTEROP
```

FUNCTION RETURN

Return value is TRUE if logon device is the console, FALSE otherwise.

PARAMETERS

None.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

None.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

MAX'DST'SIZE

PURPOSE

Returns the maximum size initially specified for a DST.

SYNTAX

```
                I                              IV
size = MAX'DST'SIZE(dst)  O-P
```

FUNCTIONAL RETURN

The maximum size for the data segment in words.

PARAMETERS

dst (Integer by value, required)
The DST number of the data segment to be examined.

CONDITION CODES

CCL Invalid DST number.
CCE Successful.
CCG DST is unassigned.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

MOUNT

PURPOSE

To establish a logical connection (mount) a Private Volume set and return an identifying handle for use by other procedures.

SYNTAX

```
                    LA      LA      LA      I
MOUNT (vsname, vsgroup, vsaccnt, reqtype,

                    IV      I       IV
       gen, pvinfo, different'pin) O-V, O-P
```

PARAMETERS

vsname (Logical Array, required)
Specifies the volume set volume name to be mounted. For example, **PVSYS**.PUB.SYS. Asterisk (*) denotes home volume set of user.

vsgroup (Logical Array, required)
Specifies the volume set group name to be mounted. For example, PVSYS.**PUB**.SYS.

vsaccnt (Logical Array, required)
Specifies the volume set account name to be mounted. For example, PVSYS.PUB.**SYS**.

reqtype (Integer, required)
An integer code specifying the type of mount to be performed. The different request codes, actions, and examples of users are as follows:

Code	Action(User)
-3	System Mount (Conditional), Binding (PVOPEN, IMPLICIT'MNT)
-2	System Mount (Conditional), No Binding
-1	Unused
0	User Mount, Binding (CXMOUNT if * specified)
1	User Mount, No Binding (CXMOUNT if other than * specified)
2	System Mount, Binding (FOPEN)
3	System Mount, No Binding (NEW/ALT/PURGEGROUP ALT/PURGEACCT REPORT)
4	Operator Mount, No Binding (LMOUNT)

gen (Integer by Value, optional)
Generation number of volume set.

Copyright (C) 1989 Propellerhead Publishing

MOUNT

pvinfo (Integer, required)
Private volume information returned by MOUNT when the volume set is mounted and possibly bound successfully. The format of *pvinfo* is as follows:

```
PVINFO.(0:8)      Volume Mask, indicating
                  volumes of set mounted.
PVINFO.(8:8)      Index into the Mounted Volume
                  Table.  This number is passed
                  on to other procedures, such
                  as DIRECSCAN.  This can also
                  be passed to the complementary
                  procedure DISMOUNT.
```

different'pin (Integer by Value, optional)
Passed by some system callers to indicate MOUNT is being performed on another process's behalf. Zero indicates mount is being done for calling process.

CONDITION CODES

CCE Successful
CCG Duplicate Mount
CCL Other Error

SPECIAL CONSIDERATIONS

Requires Privileged Mode.

DISCUSSION

Complements the procedure DISMOUNT.

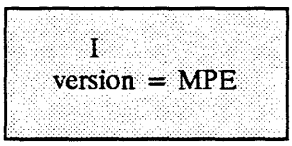
Copyright (C) 1989 Propellerhead Publishing

MPE

PURPOSE

Indicates the version of MPE in use.

SYNTAX



FUNCTION RETURN

An integer value as follows:

Value	MPE Version
0	Undefined.
1	MPE C.
4	MPE III, IV or V/P
5	MPE V/E
41	MPE XL

PARAMETERS

None.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

None.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

MUSTOPEN

PURPOSE

In normal user mode, MUSTOPEN functions the same as FOPEN. In privileged mode, if MUSTOPEN is called against a file which has a lockword, the open will succeed and the user is not prompted for the password.

SYNTAX

```
 I                          BA       LV       LV      IV
fnum = MUSTOPEN(filename, foptions, aoptions, recsize,
                           BA   BA        IV       IV
                device, formmsg, userlabels, blkfactor,
                            IV       DV     IV       IV
                numbufs, filesize, numexts, initalloc,
                            IV
                filecode)   O-V
```

FUNCTION RETURN

The file number used by the file intrinsics.

PARAMETERS

The parameters are the same as those described for the FOPEN intrinsic.

CONDITION CODES

CCL Request denied, use the FCHECK intrinsic to obtain the error code.
CCE Successful.
CCG Not returned.

SPECIAL CONSIDERATIONS

Does not require privileged mode unless the process does not want the user prompted for the file lockword.

DISCUSSION

None.

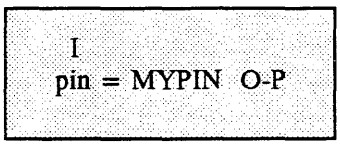
Copyright (C) 1989 Propellerhead Publishing

MYPIN

PURPOSE

Obtains the Process Identification Number (PIN) assigned to the calling process.

SYNTAX



FUNCTIONAL RETURN

The MYPIN procedure returns an integer value representing the PIN number associated with the calling process.

PARAMETERS

None.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

It is also possible to obtain one's PIN by using the following:

```
MPE-IV ->  pin := (ABSOLUTE(4) - ABSOLUTE(3)) / 16;
MPE-V  ->  pin := (ABSOLUTE(4) / 21);
```

Either method requires privileged mode. However, this method would eliminate a PCAL.

Copyright (C) 1989 Propellerhead Publishing

OBTAIN

PURPOSE

Provides a method of managing (locking) access to a resource.

SYNTAX

```
 I                    LV     LV
pin = OBTAIN(resource, alt'res)  O-P
```

FUNCTIONAL RETURN

Returns the PIN of the calling process.

PARAMETERS

resource (Logical by value, required)
A DB-relative address of the first word in the data structure managing the resource to be obtained. The entry must be initialized to zero for the first call.

alt'res (Logical by value, required)
If *resource* is locked by another process this parameter describes another resource which is to be released prior to the caller going into a wait state. This parameter would contain a DB-relative address also. A value of -1 (%177777) indicates no other resource is to be released.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode. Upon entry, DB must be pointing to the data segment containing the area used for resource management.

The resource management data structure will be formatted as follows:

Copyright (C) 1989 Propellerhead Publishing

MPE V/E

Owner PCB
PCB at Queue Head
PCB at Queue Tail
Queue Length

MPE IV or MPE V/P

Owner PIN	Queue Len
Head PIN	Tail PIN

DISCUSSION

The caller should call SETCRITICAL prior to calling OBTAIN and RESETCRITICAL upon returning. This insures the caller's priority is reduced to its original value should it have been bumped up by OBTAIN due to a lock held by a higher priority process.

The RELEASE procedure compliments OBTAIN.

Copyright (C) 1989 Propellerhead Publishing

PCB'INDEX

PURPOSE

Returns the PCB relative index for the specified PIN. Also indicates the validity of the PIN and if the process is alive.

SYNTAX

```
  I                    IV   I
result := PCB'INDEX(pin, pcbptr)  O-P
```

FUNCTIONAL RETURN

The integer value returned is as follows:

Value	Meaning
0	Successful.
1	PIN is unassigned.
2	Invalid PIN number.

PARAMETERS

pin (Integer by value, required)
Process Identification Number (PIN) of the desired process or zero (0) for the calling process.

pcbptr (Integer, required)
The PCB relative index returned for the specified PIN.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode. Split-stack calls permitted.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

PHYSICALCST

PURPOSE

Returns the CST number assigned to the specified segment.

SYNTAX

```
 I                          IV   LV
cst = PHYSICALCST(pin, segment)  O-P
```

FUNCTIONAL RETURN

The PHYSICALCST procedure returns the CST number assigned to the specified segment.

PARAMETERS

pin (Integer by value, required)
The PIN number identifying the process that is associated with the segment.

segment (Logical by value, required)
The type and physical number associated with the segment to be identified constructed as follows:

Bits	Value
0:3	The segment type: 0 - Program segment 1 - System SL segment 2 - PUB SL segment 3 - Group SL segment 4 - Program segment
8:8	The logical segment number.

CONDITION CODES

CCL Invalid PIN number.
CCE Successful.
CCG Invalid segment for specified PIN.

Copyright (C) 1989 Propellerhead Publishing

PHYSICALCST (continued)

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

PINJOBMAP

PURPOSE

Given a PIN, this procedure returns the corresponding job number and an array containing the formatted job number.

SYNTAX

```
                    IV     BA   I
PINJOBMAP(pin, job'array, job)  O-P
```

PARAMETERS

pin (Integer by value, required)
The PIN number associated to the process to be identified.

job'array (Byte array, required)
A byte array in which the formatted job number is returned. The format is "#Snnnn" or "#Jnnnn".

job (Integer, required)
The job number. If negative, the job number is for a batch (job) process. If positive, the job number is for a session.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

PROCESS'CODE'LOAD'INFO

PURPOSE

Provides access to the extension area of the Cold Load Information Table.

SYNTAX

```
  L                                          IV   IV   IV     IV
status = PROCESS'CODE'LOAD'INFO(func, word, parm1, parm2)  O-P
```

FUNCTIONAL RETURN

The return value is as follows:

Value	Meaning
0	Successful.
1	Illegal function.
2	Invalid word location.
3	Invalid parm1.

PARAMETERS

func (Integer by value, required)
A value of zero (0) to read, one (1) to write.

word (Integer by value, required);
The address in the Cold Load Information Table to be acted upon.

parm1 (Integer by value, required)
The contents of the address returned (read) or value to place in the address (write).

parm2 (Integer by value, required)
Currently not used.

CONDITION CODES

Unchanged.

Copyright (C) 1989 Propellerhead Publishing

PROCESS'CODE'LOAD'INFO (continued)

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

PROCFILE

PURPOSE

Obtain the name of the program file name currently being executed by a particular process.

SYNTAX

```
                IV  BA
PROCFILE(pin, procname)  O-P
```

PARAMETERS

pin (Integer by value, required)
A word identifier supplying the PIN number to which the call applies.

procname (Byte array, required)
An array into which the name of the program/process being execute by the specified PIN is returned. This array should be at least 27 bytes long.

CONDITION CODES

CCG The specified PIN is a system process.
CCL Invalid PIN number.
CCE Successful.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

Recommend using the supported JOBINFO intrinsic.

Copyright (C) 1989 Propellerhead Publishing

PRODUCEPARMS

PURPOSE

Parses fully-qualified name and sets up the parameters for the procedure DIRECSCAN.

SYNTAX

```
 I                                IV   BA    LA   BA
flag = PRODUCEPARMS(leaf, name, result, delim,
                                  I
                     error)   O-P
```

FUNCTIONAL RETURN

A TRUE value indicates successful completion. A FALSE value indicates an error occurred.

PARAMETERS

leaf (Integer by value, required)
Indicates the type of input contained in the ^Fcname^S parameter. If the parameter is empty, the value in parens is returned:

Type	Meaning		
0	File	F[.G[.A]]:	(@.LGN.LAN)
1	Group	G[.A]:	(@.LAN)
2	Account	A:	(@)
3	User	U[.A]:	(@.LAN)
4	Volume	V[.G[.A]]:	(@.LGN.LAN)

name (Byte array, required);
An array containing the search item.

result (Logical array, required)
A 31-word array containing the parsed parameters to be used by the 'receip' procedure in calling DIRECSCAN.

Copyright (C) 1989 Propellerhead Publishing

(0)	Index	(%0)
(2)	Type	(%2)
(3)	D'File/D'Volume Name	(%3)
(7)	D'Group/D'User Name	(%7)
(11)	D'Account Name	(%13)
(15)	D'Lockword	(%17)
(19)	G'File/G'Volume Name	(%23)
(23)	G'Group/G'User Name	(%27)
(27)	G'Account Name	(%33)

"D'xxx" contains the names used by the DIRECSCAN procedure. Must be in one of the following forms:

F.G.A @.@.A
@.G.A @.@.@

"G'xxx" contains the generic names actually specified. These can used by the *RECIP* procedure in DIRECSCAN to determine a match during the directory search.

delim (Byte array, required)
An array which will contain the final delimiter.

error (Integer, required)
Command interpreter error number associated with errors in parsing the input name byte array.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

The array generated by this procedure is used by DIRECSCAN. See the procedure DIRECSCAN for a breakdown of word 2, type, in the *result* array.

Copyright (C) 1989 Propellerhead Publishing

QUEUEPROC

PURPOSE

Used to add or remove a process from the dispatch queue.

SYNTAX

```
                                IV      IV      IV
QUEUEPROC(pcbaddr, function, location)  O-P
```

PARAMETERS

pcbaddr (Integer by value, required)
The PCB address of the process to be acted upon.

function (Integer by value, required)
= 0 If to remove the process from the dispatch queue.
= 1 If to add the process to the dispatch queue.

location (Integer by value, required)
= 0 If to place the process at the end of the processes of equal priority.
= 1 If to place the process at the beginning.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

Potential system failures include:

```
608 - Invalid PCB index.
```

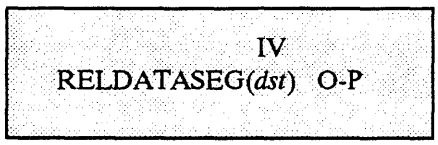
Copyright (C) 1989 Propellerhead Publishing

RELDATASEG

PURPOSE

Releases a data segment.

SYNTAX



PARAMETERS

dst (Integer by value, required)
The DST number associated with the data segment to be released.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

This procedure is used to release data segments acquired by the GETDATASEG procedure. It does not remove any references to the data segment that may exist in the process's stack or JDT if the data segment was obtained by the GETDSEG intrinsic. If this should happen, unpredictable results could occur. System failures will occur is a the DST is a system table, locked or frozen in memory. The following are the system failures which can be caused by improper use of RELDATASEG:

```
124 - Attempting to release a system data segment.
630 - Attempting to release a resident segment.
631 - Attempting to release a segment frozen for I/O.
632 - Unable to release virtual memory for the segment.
634 - PDISABLE count > 0.
```


Copyright (C) 1989 Propellerhead Publishing

RELEASE

PURPOSE

Unlocks a resource previously locked via OBTAIN.

SYNTAX

```
                    LV    LV    LV
RELEASE(resource, altres, wakeup) O-P
```

PARAMETERS

resource (Logical by value, required)
The DB relative address of the resource to be released.

altres (Logical by value, required)
Contains a DB relative address for an alternate resource to transfer all waiters of the current resource being released. A value of -1 (%177777) indicates no transfer is to be performed.

wakeup (Logical by value, required)
If zero (0), all processes queued for the specified resource are launched. A value of one (1) causes only the first process on the queue to be launched.

If a value of zero (0) is specified, the highest priority process will become the next owner of the resource. However, all processes will by launched. This may cause considerable overhead since all segments of each process will have to be brought into memory only to have the processed blocked again. When using this value, the calling process should check the data structure and ensure it is the resource owner.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode. Must be called DB pointing to the data structure used for resource management.

DISCUSSION

Compliments the OBTAIN procedure.

Copyright (C) 1989 Propellerhead Publishing

RELSIR

PURPOSE

Releases the system internal resource (SIR) that was obtained by a call to GETSIR.

SYNTAX

```
          LV LV
RELSIR(sir, flag)  O-P
```

PARAMETERS

sir (Logical by value, required)
This word identifies the SIR being released (see GETSIR).

flag (Logical by value, required)
This word contains the flag returned by the GETSIR procedure when the specified SIR was first locked.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode. Multiple SIRs must be released in LIFO fashion. It may also be necessary to enable 'BREAK' after releasing all locked SIRs or remove the calling process from CRITICAL mode (see GETSIR).

DISCUSSION

This procedure and it's complement procedure GETSIR are used to lock a system internal resource number (SIR) which is associated with an internal resouce to prevent concurrent access.

Copyright (C) 1989 Propellerhead Publishing

REMOVE'PASSWORDS

PURPOSE

Removes passwords/lockwords from character strings.

SYNTAX

```
                                    BA I
REMOVE'PASSWORDS(buffer,len)
```

PARAMETERS

buffer (Byte array, required)
The byte array containing the text which may have passwords or lockwords which are to be removed. To be removed, the password/lockword must have the following characteristics:

Value	Meaning
1	Begins with a slash.
2	The first character is alpha.
3	The password/lockword is delimited by either a period, comma, semicolon, space or carriage return.
4	The password/lockword is no longer than 8 characters.

len (Integer, required)
The length of the buffer. Upon return, this will contain the new length of the buffer.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

None.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

REQUESTSERVICE

PURPOSE

Determines if break was hit or if the process has an abort pending.

SYNTAX

```
   L
break = REQUESTSERVICE  O-P
```

FUNCTIONAL RETURN

Return value TRUE if break was hit (and was enabled) or if the process has an abort request pending, FALSE otherwise.

PARAMETERS

None.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

RESETBREAKBITS

PURPOSE

Procedure to reset BREAK and CONTROL-Y bits in the LPDT for the specified device.

SYNTAX

```
 L                                    IV   LV
states = RESETBREAKBITS(device, flag)  O-P
```

FUNCTIONAL RETURN

This procedure returns a TRUE/FALSE indicating whether or not the specified flag was set before clearing.

PARAMETERS

device (Integer by value, required)
The number of the logical device on which the function is to be performed.

flag (Logical by value, required)
This parameter indicates which bit is to be reset. A value of TRUE will cause the BREAK bit to be cleared, a FALSE value will cause the CONTROL-Y bit to be reset.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

RESETCRITICAL

PURPOSE

Reverses the condition established by the SETCRITICAL procedure. Any pending aborts on the process can then take place.

SYNTAX

PARAMETERS

critical (Logical by value, required)
This is the value returned by the SETCRITICAL procedure when the calling process was placed in critical mode. It is used to keep track of nested calls to SETCRITICAL.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode. Also, after returning from critical mode execution the arithmetic traps can be enabled as follows:

```
PUSH(STATUS);
TOS.(2:1) := 1;
SET(STATUS);
```

DISCUSSION

This procedure and it's complement procedure SETCRITICAL are used to insure a series of code is performed without interruption.

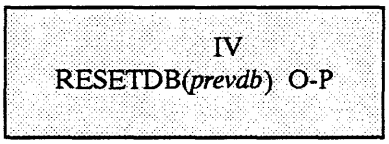
Copyright (C) 1989 Propellerhead Publishing

RESETDB

PURPOSE

Sets the DB register to a specific address in bank 0, returns to the split-stack DST or process's stack.

SYNTAX



PARAMETERS

prevdb (Integer by value, required)
This is the value returned by the SETSYSDB procedure when the calling process's DB register was set to the system global DB. If this parameter is a negative 1 (-1), the DB register is set to the address contained in the absolute DB address of the process's PCB. If this value is zero, DB is reset to the process's stack. If the value is positive, DB is set to the address in bank 0 but the PCB will not reflect this setting.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

Compliments the SETSYSDB procedure. Potential system failures include:

```
611 - Illegal address.
```

Copyright (C) 1989 Propellerhead Publishing

RLOCK

PURPOSE

Obtain a lock on the specified RIN.

SYNTAX

```
                IV    LV
RLOCK(rin, condition)  O-P
```

PARAMETERS

rin (Integer by value, required)
The number of the RIN to be locked. If bit 0 of this parameter is on it indicates that the DB register of the calling process is already pointing to the RIN table.

condition (Logical by value, required)
If TRUE, then lock the RIN unconditionally. If the RIN is already locked by some other process, the calling process will be placed in the appropriate RIN wait. If FALSE, then lock the RIN only if it is available. If some other process already has the RIN, return to the calling process.

CONDITION CODES

CCL RIN already locked by another process.
(Returned only if condition=FALSE)
CCE Successful.
CCG Calling process already has RIN locked.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

User is not required to have MR capability to lock multiple RINs. See the RUNLOCK procedure to release any RINs acquired by RLOCK.

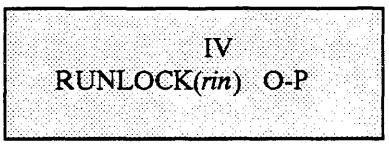
Copyright (C) 1989 Propellerhead Publishing

RUNLOCK

PURPOSE

Unlocks the specified RIN.

SYNTAX



PARAMETERS

rin (Integer by value, required)
The number of the RIN to be unlocked. If bit 0 of this parameter is on it indicates that the DB register of the calling process is already pointing to the RIN table.

CONDITION CODES

CCL The specified RIN was not allocated.
CCE Successful.
CCG The calling process did not have the RIN locked.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

Compliments the RLOCK procedure.

Copyright (C) 1989 Propellerhead Publishing

SCHEDDAYTIME

PURPOSE

Formats the specified date and time into a 14 character string.

SYNTAX

```
                                BA  IA
SCHEDDAYTIME(buffer, time)      O-P
```

PARAMETERS

buffer (Byte array, required)
The date and time returned in a the following 14 character format:

MM/DD/YY HH:SS

time (Integer array, required)
An array containing the following:

Word	Content
0	The date in the same format as returned by the CALENDAR intrinsic.
1-2	The time in the same format as returned by the CLOCK intrinsic.

CONDITION CODES

CCL Not returned.
CCE Successful.
CCG Conversion error.

SPECIAL CONSIDERATIONS

Requires privileged mode. Do not call in split-stack mode.

DISCUSSION

None.

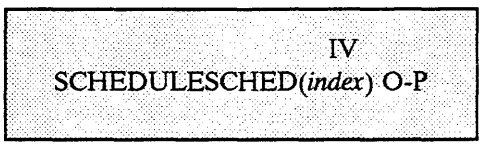
Copyright (C) 1989 Propellerhead Publishing

SCHEDULESCHED

PURPOSE

Links a JMAT entry into the appropriate slot in the schedule job queue.

SYNTAX



PARAMETERS

index (Integer by value, required)
The address in the JMAT table of the job which is to be linked into the schedule queue. The job must be in the INTRO state.

CONDITION CODES

Not returned.

SPECIAL CONSIDERATIONS

Requires privileged mode. Split stack calls not permitted.

DISCUSSION

If the job start time on the JMAT entry is less than the current time, the job will be placed at the head of the queue. When UCOP is activated, the job will be initiated, or if no batch job positions are available, the job will be placed in the WAIT state.

Note, the calling process is responsible for locking the JMAT SIR.

Copyright (C) 1989 Propellerhead Publishing

SECHK'PASSWORD

PURPOSE

Determines if a password is valid for the specified user and account names.

SYNTAX

```
                                          I                                   BA      BA      BA
error = SECHK'PASSWORD(password, name1, name2,
                                                                              L
                                                                    encrypted) O-P
```

FUNCTIONAL RETURN

A zero (0) indicates successful, non-zero indicates an error occurred.

PARAMETERS

password (Byte array, required)
An array containing the password which is to be verified. The password must not exceed 8 characters and cannot begin with a numeric character. It also must be left justified, upshifted and padded with trailing blanks.

name1 (Byte array, required);
An array containing either the user, group or account name depending on the password being examined. This also must be left justified, upshifted and padded with trailing blanks.

name2 (Byte array, required)
An array containing the account name. If an account password is to be examined, this parameter should contain the word "SYSTEM".

encrypted (Logical, required)
A logical value indicating whether the password was encrypted (TRUE) or not (FALSE).

CONDITION CODES

Not returned.

Copyright (C) 1989 Propellerhead Publishing

SECHK'PASSWORD (Continued)

SPECIAL CONSIDERATIONS

Requires privileged mode. Also, this is a PASCAL procedure and only callable from SPL or PASCAL due to the parameter checking level.

DISCUSSION

This procedure will validate a user, group or account password. If the actual password is encrypted, *name1* and *name2* are used to obtain the encryption seed and the content of *password* is encrypted prior to the comparison. If encryption is enabled, the contents of *password* will be returned encrypted.

See GETSALT and CRYPT for additional information.

Copyright (C) 1989 Propellerhead Publishing

SECURITYINSTALLED

PURPOSE

Indicates if HP's security product is installed.

SYNTAX

```
L
installed = SECURITYINSTALLED
```

FUNCTION RETURN

Return value is TRUE if the security product is installed, FALSE otherwise.

PARAMETERS

None.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

None.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

SEGMENTER

PURPOSE

Provides programmatic access to the SEGMENTER procedure.

SYNTAX

```
                    I    IV  I   IV IV IV IV IV IV
SEGMENTER (pin, code, err, p1, p2, p3, p4, p5, p6,
                   BA BA  BA     BA
           st1, st2, name1, name2)  O-V
```

PARAMETERS

pin (Integer, required)
The PIN number of the segmenter process - SEGPROC.PUB.SYS. Should be set to zero for the first call. Each subsequent call should contain the PIN number. (See the table)

code (Integer by value, required)
This is a code identifying the command to be performed. The remaining parameters for the call depend on the command being performed.

err (Integer, required)
A returned error code indicating the failure of the command when CCL or CCG is returned:

Code	Type of Error
0	Failure message printed on STDLIST
1	Warning printed on $STDLIST
2	Required parameter omitted.
3	Unable to create SEGPROC.
4	Unable to activate SEGPROC.
5	Unable to send mail to SEGPROC.
6	Unable to receive mail from SEGPROC.
7	SEGPROC aborted.

p1 (Integer by value)
See Table.

p2 (Integer by value)
See Table.

Copyright (C) 1989 Propellerhead Publishing

p3 (Integer by value)
Operational flags as follows:

Bits	Contents
15:1	List bit, set to get PMAP, (ADDSL, PREP) and also for LISTRL, LISTSL, LISTUSL, and LISTAUX commands.
14:1	ZERODB flag.
12:2	0 - operate on a segment 1 - operate on a unit 2 - operate on an entry.
11:1	Ignore file equation, if set, all output goes to $STDLIST. Otherwise it will go to SEGLIST.
10:1	If set, allows the purging of loaded segments.
9:1	NOSYM bit
8:1	FPMAP bit
7:1	NO FPMAP bit
6:1	CHECKSUM specified.
3:3	Unused.
2:1	System bit, considers the segment added to the SL a system segment.
1:1	Resident bit. Considers the segment added to the SL a memory resident segment.
0:1	Allocated bit. Considers the segment added to the SL to be permanently allocated.

p4 (Integer by value)
See table.

p5 (Integer by value)
See table.

p6 (Integer by value)
See table.

st1 (Byte Array)
See Table.

name1 (Byte Array)
See Table.

name2 (Byte Array)
See Table

p4 (Integer by value)
See Table.

p5 (Integer by value)
See Table.

Copyright (C) 1989 Propellerhead Publishing

SEGMENTER (Continued)

Command	Code	p1	p2	p3	p4	p5	p6	st1	st2	Name1	Name2
ADDRL	0	INDX	..	..	..	..	..	RBM	..	...	...
ADDSL	1	..	..	*1*	..	..	PTCH	...	SEG	...	...
AUXUSL	2	..	..	0	..	..	..	...	...	FILE	...
BUILDRL	3	RECS	EXTS	0	..	..	..	...	...	FILE	...
BUILDSL	4	RECS	EXTS	0	..	..	..	...	...	FILE	...
BUILDUSL	5	RECS	EXTS	0	..	..	..	...	...	FILE	...
CEASE	6	INDX	..	*1*	..	..	..	...	...	NAME	...
COPY	7	INDX	..	*1*	..	..	..	...	...	NAME	...
EXIT	8	..	..	0	..	..	..	...	...	...	...
HIDE	9	INDX	..	0	..	..	..	...	...	NAME	...
LISTRL	10	..	..	*1*	..	..	..	...	...	...	...
LISTSL	11	..	..	*1*	..	..	..	...	...	SEG	...
LISTUSL	12	..	..	*1*	..	..	..	...	SEG	...	...
NEWSEG	13	INDX	..	0	..	..	..	RBM	SEG	...	...
PREPARE	14	STCK	DL	*1*	MAXD	CAP	PTCH	...	...	NAME	RL
PURGERBM	15	INDX	..	*1*	..	..	..	NAME	...	...	...
PURGERL	16	..	..	*1*	..	..	..	NAME	...	...	...
PURGESL	17	..	..	*1*	..	..	..	NAME	...	...	...
REVEAL	18	INDX	..	0	..	..	..	NAME	...	...	...
RL	19	..	..	0	..	..	..	...	...	FILE	...
SL	20	..	..	0	..	..	..	...	...	FILE	...
USE	21	INDX	..	*1*	..	..	..	NAME	...	...	...
USL	22	..	..	0	..	..	..	...	...	FILE	...
DEBUG	23	..	..	0	..	..	..	...	...	...	...
COPYSL	24	%	..	0	..	..	..	...	...	FILE	...
COPYUSL	25	%	..	0	..	..	..	...	...	FILE	...
CLEANSL	26	..	..	0	..	..	..	...	...	FILE	...
CLEANUSL	27	..	..	0	..	..	..	...	...	FILE	...
LISTAUX	29	..	..	*1*	..	..	..	...	SEG	...	...
SHOW	30	..	..	0	..	..	..	...	...	...	...
LISTPMAP	31	..	..	0	..	..	..	...	...	...	...
SETFPMAP	32	..	..	0	..	..	..	...	...	...	...

INDX - Index MAXD - Maxdata
STCK - Stack PTCH - Patch
CAP - Capability

1 - See description of p3 above.

CONDITION CODES

CCL	Error occurred, SEGPROC aborted.
CCE	Successful.
CCG	Error occurred, SEGPROC still available.

SPECIAL CONSIDERATIONS

None.

Copyright (C) 1989 Propellerhead Publishing

DISCUSSION

If defaults are to be used for the parameters, use a -1 value for that parameter.

Copyright (C) 1989 Propellerhead Publishing

SET'BREAKPNT

PURPOSE

Performs programmatic setting of break points from within the calling processes.

SYNTAX

```
                                I   IV  IV   IV      LV
SET'BREAKPNT(error, seg, p'loc, segtype, break'type,
                                IV      LA     IV
                             count, break'cond, label) O-V
```

PARAMETERS

error (Integer, required)
An error code return if call was unsuccessful. Possible values are:

Value	Description
0	Successful.
1	Invalid program location.
2	Break point table is full.
3	Process/system conflict.
4	Location conflict.
5	Unused.
6	Invalid segment.
7	Invalid segment type.
8	Invalid count.
9	Insufficient capability.
10	Invalid procedure label.
11	Invalid break point type.

seg (Integer by value, optional);
The segment in which the break point is to be set. If not specified, current segment is used.

p'loc (Integer by value, required)
The PB relative location where the break point is to be placed.

seg'type (Integer by value, optional)
The type of segment number contained in the 'seg' parameter:

Value	Segment Type
0	Absolute memory segment.
1	System SL.
2	PUB SL.
3	Group SL.
4	Program segment. (Default)

Copyright (C) 1989 Propellerhead Publishing

If not specified, program segment is assumed.

break'type (Logical by value, optional)
This word describes the type of break point as follows:

Bits	Description
13:3	= 0: Local break point.
	= 1: System break point.
12:1	= 0: Temporary break point.
	= 1: Permanent break point.

count (Integer by value, optional)
Which occurrence the break point is to be executed.

break'cond (Logical array, optional)
Not currently used.

label (Integer by value, optional)
The external PLABEL of the user's trap procedure. If not supplied, DEBUG will be used.

CONDITION CODES

CCL Failure.
CCE Successful.
CCG Not returned.

SPECIAL CONSIDERATIONS

None.

DISCUSSION

Compliments the procedure CLEAR'BREAKPNT.

Copyright (C) 1989 Propellerhead Publishing

SET'PSIF

PURPOSE

Procedure to set the pseudo-interrupt flags for the specified PCB.

SYNTAX

```
                    IV    LV
SET'PSIF(pcbaddr, flag)  O-P
```

PARAMETERS

pcbaddr (Integer by value, required)
The PCB address of the process to be modified.

flag (Logical by value, required)
A value indicating which pseudo-interrupt is to be set. The following are the valid pseudo-interrupt flags:

Flag	Pseudo-Interrupt
%40	Hard kill - process killed by UCOP or QUITPROG.
%20	Soft kill - process simply calls terminate.
%2	A Control-Y interrupt.
%1	A BREAK interrupt.

Any other value will cause a system failure.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

Compliments the CLEAR'PSIF procedure. The specified PCB must be valid and alive or a system failure will occur. DISPATCHER should be disabled during this call. This procedure will place an entry on the dispatch queue for the specified PCB and request a dispatch.

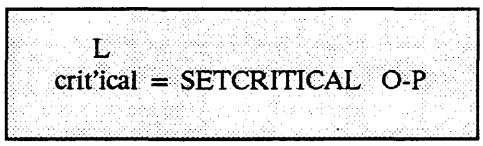
Copyright (C) 1989 Propellerhead Publishing

SETCRITICAL

PURPOSE

Prevents the current running process from being aborted during the execution of a critical series of code.

SYNTAX



FUNCTION RETURN

The logical value returned by SETCRITICAL is required by the RESETCRITICAL procedure to reverse the process and allow interrupts. It indicates whether the process was already critical. It is used by RESETCRITICAL to the process can be protected even if nested SETCRITICAL's have been used. It will have a value of TRUE if the process was previously critical, FALSE otherwise.

PARAMETERS

None.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode. Also, all arithmetic traps need to be turned off as follows:

```
PUSH(STATUS);
TOS.(2:1) := 0;
SET(STATUS);
```

DISCUSSION

This procedure and it's complement procedure RESETCRITICAL are used to insure a series of code is performed without being aborted.

A process which terminate while in critical mode will cause a system failure.

Copyright (C) 1989 Propellerhead Publishing

SETDBTOSTACK

PURPOSE

Sets the calling process's DB register to the base address of the process's stack DST.

SYNTAX

```
D
prev'db = SETDBTOSTACK;  O-P
```

FUNCTION RETURN

The double word value returned by SETDBTOSTACK is the bank and address of the DB register. This is used to reset the DB register back to is original state.

PARAMETERS

None.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

Compliment procedure is CHANGEDB.

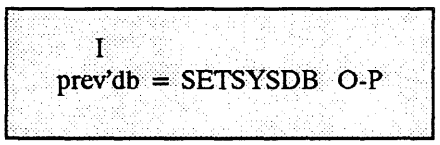
Copyright (C) 1989 Propellerhead Publishing

SETSYSDB

PURPOSE

Sets the calling process's DB register to the system global DB (%1000).

SYNTAX



FUNCTION RETURN

The integer value returned by SETSYSDB is the value of the DB register prior to the call. This is used to reset the DB register back to is original state.

PARAMETERS

None.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

Compliment procedure is RESETDB.

Copyright (C) 1989 Propellerhead Publishing

SFINDODD

PURPOSE

Returns the XDD segment relative address for the specified XDD entry.

SYNTAX

```
                L                          IV    I
error = SFINDODD(dfid, xdd'addr)  O-P
```

FUNCTIONAL RETURN

A value of TRUE if the specified XDD entry was located, FALSE otherwise.

PARAMETERS

dfid (Integer by value, required)
The XDD entry to be located, specified as follows:

Bits	Contents
0:1	0 = IDD entry number.
	1 = ODD entry number.
1:15	The XDD entry number.

xdd'addr (Integer, required)
The XDD-relative address of the specified entry as follows:

Bits	Contents
0:1	0 = IDD entry address.
	1 = ODD entry address.
1:15	The XDD-relative address.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode. Split-stack calls not permitted.

DISCUSSION

None.

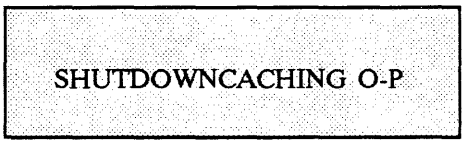
Copyright (C) 1989 Propellerhead Publishing

SHUTDOWNCACHING

PURPOSE

Stops caching on all discs.

SYNTAX



PARAMETERS

None.

CONDITION CODES

Not returned.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

This procedure is normally called during a '=SHUTDOWN'.

Copyright (C) 1989 Propellerhead Publishing

STACK'DST

PURPOSE

Obtains the DST number of the calling process' stack.

SYNTAX

```
    L
stack = STACK'DST  O-P
```

FUNCTION RETURN

Returns the DST number associated with the calling process' stack.

PARAMETERS

None.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

STACKCHECK

PURPOSE

Determines if specified DST is a process stack.

SYNTAX

```
    L                          IV
stack = STACKCHECK(dst)  O-P
```

FUNCTIONAL RETURN

A value of TRUE if the specified DST number is a stack, FALSE otherwise.

PARAMETERS

dst (Integer by value, required)
The number of the data segment to be tested.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

STARTSTATISTICS

PURPOSE

Initiates the MPE Internal Measurement Facility.

SYNTAX

```
                I                                LV
error = STARTSTATISTICS(classmask)  O-P
```

FUNCTIONAL RETURN

An error code is returned indicating a failure in starting the MPE Internal Measurement Facility as follows:

Value	Meaning
0	Successful.
1	No virtual memory available for data segment(s).
2	No DST entries available.
3	Could not lock data segment in memory.

PARAMETERS

classmask (Logical by value, required)
This is a bit mask used to identify which MPE Measurement Interface statistics are to be gathered by the facility. The following are the available classes of statistical data:

Bit	Class
0	Class 0, System global data and also disc, printer and tape usage.
14	Class 14, I/O statistics.
15	Class 15, process information.

All other bits should be set to zero.

CONDITION CODES

CCL Failed to enable MPE Measurement Facility.
CCE Successful.
CCG Not returned.

Copyright (C) 1989 Propellerhead Publishing

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

The STOPSTATISTICS procedure is used to stop specific aspects of the Measurement Interface. It complements the STARTSTATISTICS procedure.

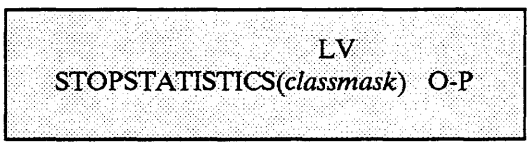
Copyright (C) 1989 Propellerhead Publishing

STOPSTATISTICS

PURPOSE

Terminates the MPE Internal Measurement Facility.

SYNTAX



PARAMETERS

classmask (Logical by value, required)
This indicates which aspects of the facility are to be turned off. Each bit represents a class of statistics as follows:

Bit	Class
0	Class 0, global data and disc, printer and tape usage.
14	Class 14, I/O statistics.
15	Class 15, process data.

All remaining bits should be set to zero.

CONDITION CODES

CCL Failure.
CCE Successful.
CCG Not returned.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

The gathering of statistics for the specific class will only stop once the user count falls to zero.

Compliments the STARTSTATISTICS procedure.

Copyright (C) 1989 Propellerhead Publishing

STRINGPINATTAIL

PURPOSE

Places the specified PIN at the end of the list of impeded processes strung together via PCB entries.

SYNTAX

```
                                    IV      IV
STRINGPINATTAIL(headpin, addpin)  O-P
```

PARAMETERS

headpin (Integer by value, required)
The PIN number of the first process currently on the waiting list.

addpin (Integer by value, required)
The PIN of the process to be added to the tail of the waiting list.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode. Split-stack calls are permitted.

DISCUSSION

The caller is responsible for disabling pseudo and external interrupts. Also, this procedure does not affect the process' wait state.

Copyright (C) 1989 Propellerhead Publishing

SYSGLOB

PURPOSE

Returns a specified word from the System Global Area or the System Global Extension Area.

SYNTAX

```
                I                    IV
value = SYSGLOB(address)
```

FUNCTIONAL RETURN

The contents of the specified System Global or System Global Extension Area.

PARAMETERS

address (Integer by value, required)
The System Global relative address of the word to be returned. Addresses in the range %0 - %377 refer to Sysglob, while %400-- %577 refer to the SysGlobExt.

CONDITION CODES

CCL Invalid address.
CCE Successful.
CCG Not returned.

SPECIAL CONSIDERATIONS

None.

DISCUSSION

Split-stack calls are permitted.

Copyright (C) 1989 Propellerhead Publishing

SYSPROC

PURPOSE

Converts a PIN number to a PCB index.

SYNTAX

```
        L                    IV
pcb'index = SYSPROC(pin)  O-P
```

FUNCTIONAL RETURN

The PCB relative address of the specified PIN in the PCB table.

PARAMETERS

pin (Integer by value, required)
The PIN number associated with the process whose entry address is to be returned.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

THISCPU

PURPOSE

Obtain a code representing the type of computer on which the program is being executed.

SYNTAX

```
        I
cpu'type = THISCPU
```

FUNCTION RETURN

This integer value returned by THISCPU represents the following:

Series	THISCPU
I	0
II	1
30, 33	2
III	3
4x, 5x	4
64, 68, 70	5
37	6
Micro 3000 X/E	7
Micro 3000	8
MPE-XL	16

PARAMETERS

None.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

None.

Copyright (C) 1989 Propellerhead Publishing

THISCPU (continued)

DISCUSSION

Unlike the machine instruction PCN, this procedure does not require privileged mode. The values returned by this procedure are different than those returned from the machine instruction.

Potential system failures include:

```
13 - Unknown value returned from microcode.
```


Copyright (C) 1989 Propellerhead Publishing

TIMEREQ

PURPOSE

Sets up a time to expire after a specified interval.

SYNTAX

```
                                I                    IV   IV   DV
trlindex = TIMEREQ(code, reqid, time)  O-P
```

FUNCTIONAL RETURN

The index of this entry into the TRL.

PARAMETERS

code (Integer by value, required)
A code indicating the type of request (see *reqid*).

reqid (Integer by value, required)
The meaning is dependent upon the value of *code* as follows:

Code	Type	ReqID	Action
5	Pause/Delay	PIN	UNIMPEDE(pin)
8	Terminal Timeout	Port Mask	AWAKE(ucop'pin)
10	PCB WatchdDog	PIN	AWAKE(pin)
11	Port Timeout	Timer Port DST	SEND(port'id)
other	I/O Timeout	DIT Pointer	AWAKEIO(dit'ptr)

time (Double by value, required)
The number of milliseconds from now when the timer is to expire. Resolution is 100ms.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

Copyright (C) 1989 Propellerhead Publishing

DISCUSSION

Potential system failures include:

```
3 - Out of TRL entries.
25 - TRL free list pointer invalid.
26 - Invalid reqid
```


Copyright (C) 1989 Propellerhead Publishing

TOCALENDAR

PURPOSE

Given year, month and day, determined day of year.

SYNTAX

```
                                IV   IV   IV    I
TOCALENDAR(year, month, day, day'of'year)
```

PARAMETERS

year (Integer by value, required)
The year (i.e. 1988).

month (Integer by value, required)
The month of the year (1 = January, 12=December) returned.

day (Integer by value, required)
The day of month.

day'of'year (Integer, required)
The day of year returned.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

None.

DISCUSSION

See the procedure FROMCALENDAR.

Copyright (C) 1989 Propellerhead Publishing

UDC

PURPOSE

Executes User Defined Commands (UDCs).

SYNTAX

```
L              BA      IA
result = UDC(command, offset) O-P
```

FUNCTIONAL RETURN

Returns a value of TRUE if *command* was a UDC command, FALSE otherwise.

PARAMETERS

command (Byte array, required)
An array containing the UDC command terminated with a carriage return (%15).

offset (Integer by value, required)
An index into the UDC directory to allow nesting but prevent recursion.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

UNCACHE'LDEV

PURPOSE

Disables caching for the specified disc drive.

SYNTAX

```
                                    IV   IV
UNCACHE'LDEV(ldev, status)  O-P
```

PARAMETERS

ldev (Integer by value, required)
The logical device number of the disc drive on which caching is to be turned off.

status (Integer by value, required)
A returned value indicating the status of the call. The possible values are:

Value	Meaning
0	Successful.
1	Caching was not enabled.
2	Internal cache error occurred.
3	The disc drive specified did not have caching enabled.
4	(Not used).
5	Device is not cachable.
6	Caching is not installed.

CONDITION CODES

Not returned.

SPECIAL CONSIDERATIONS

Requires privileged mode. Do not call in split-stack mode.

DISCUSSION

Also see the procedure CACHE'LDEV.

Copyright (C) 1989 Propellerhead Publishing

UNFREEZESEG'

PURPOSE

Unfreezes the specified DST from memory and allow memory manager from relocating it in memory.

SYNTAX

```
                    IV
UNFREEZESEG'(seg'id) O-P (MPE-IV)

                    DV
UNFREEZESEG'(seg'id) O-P (MPE-V)
```

PARAMETERS

seg'id (Logical by value, required)
The number of the segment to be unfrozen from memory composed as follows:

(MPE-IV)

Bits	Description
0:1=1	Segment is part of program 1:7 - CSTXBLK INDEX, 8:8 - Segment number(1-255)
0:2=0	Segment is a DST, 2:14 - DST number.
0:2=1	Segment is a SL Segment, 2:14 - SL Segment Number.

(MPE-V)

Word	Bits	Description
0	0:3	Type: 0 = Data Segment 1 = SL Segment 2 = Program Segment 3 = Cache Domain
	4:12	Program CSTXBLK index.
1	0:16	DST, CST, CSTX or CDT Number

CONDITION CODES

Unchanged.

Copyright (C) 1989 Propellerhead Publishing

UNFREEZESEG' (continued)

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

Compliments the procedure FREEZESEG'. This is an extra entry point into the LOCKSEG' procedure. This procedure decrements the freeze count in the segment's region header. When this count reaches zero, the segment becomes available to memory manager.

Potential system failures include:

```
606 - The segment is not frozen, the freeze count is
      negative or the segment is absent.
```

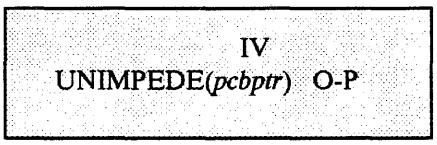
Copyright (C) 1989 Propellerhead Publishing

UNIMPEDE

PURPOSE

Unimpedes the process located at the specified address in the PCB table.

SYNTAX



PARAMETERS

pcbptr (Integer by value, required)
The PCB relative base address of the process to be unimpeded.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

Compliments the procedure IMPEDE.

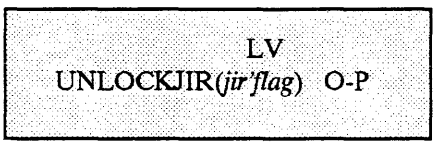
Copyright (C) 1989 Propellerhead Publishing

UNLOCKJIR

PURPOSE

Releases the process's JIR.

SYNTAX



PARAMETERS

jir'flag (Logical by value, required)
This word contains the flag returned by the LOCKJIR procedure when the specified JIR was first locked.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

This procedure complements LOCKJIR.

Copyright (C) 1989 Propellerhead Publishing

UNLOCKSEG

PURPOSE

Unlocks the specified segment to allow it to be swapped to virtual storage.

SYNTAX

```
                LV  LV LV
UNLOCKSEG(seg, flag, pin)  O-P (MPE-IV)

                DV  LV LV
UNLOCKSEG(seg, flag, pin)  O-P (MPE-V)
```

PARAMETERS

seg (Logical by value, required)
A word identifying the segment to be unlocked from memory:

(MPE-IV)

Bits	Description
0:1=1	Segment is part of program 1:7 - CSTXBLK INDEX, 8:8 - Segment number(1-255)
0:2=0	Segment is a DST, 2:14 - DST number.
0:2=1	Segment is a SL Segment, 2:14 - SL Segment Number.

(MPE-V)

Word	Bits	Description
0	0:3	Type: 0 = Data Segment 1 = SL Segment 2 = Program Segment 3 = Cache Domain
	4:12	Program CSTXBLK index.
1	0:16	DST, CST, CSTX or CDT Number

flag (Logical by value, required)
This flag indicates the type of segment to be unlocked:

TRUE - The segment identifier 'seg' contains a DST number.

Copyright (C) 1989 Propellerhead Publishing

```
FALSE -The segment identifier 'seg' is either a CST
       number if PIN = zero or a CST block index
       relative to %300 if PIN not equal to zero.
```

pin (Logical by value, required)
If locking a CST block index, this contains the PIN number of the process. Otherwise zero.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

Compliments the procedure LOCKSEG.

Potential system failures include:

```
606 -  The segment is not locked, lock count is
       negative or the segment is absent.
```

Copyright (C) 1989 Propellerhead Publishing

UNLOCKSEG'

PURPOSE

Unlocks the specified DST to allow it to be swapped to virtual storage.

SYNTAX

```
                    LV
UNLOCKSEG'(seg'id)  O-P (MPE-IV)

                    DV
UNLOCKSEG'(seg'id)  O-P (MPE-V)
```

PARAMETERS

seg'id (Logical by value, required)
The number of the segment to be unlocked from memory composed as follows:

(MPE-IV)

Bits	Description
0:1=1	Segment is part of program 1:7 - CSTXBLK INDEX, 8:8 - Segment number(1-255)
0:2=0	Segment is a DST, 2:14 - DST number.
0:2=1	Segment is a SL Segment, 2:14 - SL Segment Number.

(MPE-V)

Word	Bits	Description
0	0:3	Type: 0 = Data Segment 1 = SL Segment 2 = Program Segment 3 = Cache Domain
	4:12	Program CSTXBLK index.
1	0:16	DST, CST, CSTX or CDT Number

CONDITION CODES

Unchanged.

Copyright (C) 1989 Propellerhead Publishing

UNLOCKSEG' (continued)

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

Compliments the procedure LOCKSEG'.

Potential system failures include:

```
606 -  The segment is not locked, lock count is
       negative or the segment is absent.
```

Copyright (C) 1989 Propellerhead Publishing

UNSTRINGHEADPIN

PURPOSE

Removes the specified PCB entry from the string of PCB entries on a waiting (impeded) list.

SYNTAX

```
                I                                IV
newhead = UNSTRINGHEADPIN(pin) O-P
```

FUNCTIONAL RETURN

Returns the PIN of the process which will be the new head of the list. A zero (0) indicates this was the last PIN on the list.

PARAMETERS

pin (Integer by value, required)
The PIN of the process to be removed from the waiting (impede) list.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode. Split-stack calls permitted.

DISCUSSION

The caller is responsible for disabling pseudo and external interrupts. Also, this procedure does not change the process' wait state.

Copyright (C) 1989 Propellerhead Publishing

VERIFY'PASSWORD

PURPOSE

Verifies the match of a password with that contained in the directory entry. Since password encryption may be enabled, the procedure may have to encrypt the password before making the comparison.

SYNTAX

```
                                                  BA    IV    LA    BA
L
match = VERIFY'PASSWORD(password, level, entry, name,

                                                  BA
                                               rootname)  O-P
```

FUNCTIONAL RETURN

A TRUE value indicates successful password is valid, FALSE indicates password doesn't match.

PARAMETERS

password (Byte array, required)
An array containing the password to be tested.

level (Integer by value, required);
An integer indicating what type of password is being verified:

Level	Meaning
1	Group password.
2	Account password.
3	User password.

entry (Logical array, required)
A logical array containing the group, account or user directory entry which is consistent with *level.*

name (Byte array, required)
Contains the name of the group, account or user whose password is being verified. The name must be upper case, left justified and padded with trailing blanks.

rootname (Byte array, required)
The name of account or the constant "SYSTEM". This is used to generate the salt for the encryption procedure.

Copyright (C) 1989 Propellerhead Publishing

VERIFY'PASSWORD(continued)

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

To determine if a password is encrypted, the following flags exist in the appropriate directory entries:

Entry Type	Word	Flag
Account	%33	2:1 - Password Encrypted.
		3:1 - Password Required.
Group	%25	1:1 - Password Encrypted.
User	%21	2:1 - Password Encrypted.
		3:1 - Password Required.
		4:1 - Password Expired.

Copyright (C) 1989 Propellerhead Publishing

VTABINX

PURPOSE

Return the volume table index assigned to the specified disc device.

SYNTAX

```
                I                          IV    IV
vtable'index = VTABINX(ldev, voltabx)  O-P
```

FUNCTIONAL RETURN

The volume table index associated with the specified disc device.

PARAMETERS

ldev (Integer by value, required)
The logical device number of the disc.

voltabx (Integer by value, required)
The mounted private volume table index, zero if only system discs available.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

Many of the double word disc addresses that are used for directory calls or stored in various system tables actually contain a volume index in the first 8 bits, not a logical device number. This routine converts the logical device number to the volume table index.

Copyright (C) 1989 Propellerhead Publishing

VTABTOLDEV

PURPOSE

Converts a disc file extent map from volume table indexes to logical device numbers.

SYNTAX

```
                                DA    DA   IV      IV
VTABTOLDEV(ext'ldev, ext'vol, count, mvtabx)  O-P
```

PARAMETERS

ext'ldev (Double array, required)
The returned double-word extent map containing logical device numbers.

ext'vol (Double array, required)
The source double-word extent map containing volume table indexes.

count (Integer by value, required)
The number of extent entries in *ext'vol*.

mvtabx (Integer by value, required)
The mounted volume table index for if file located on private volume or zero (0) if system domain.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode. Cannot be called in split-stack.

DISCUSSION

Process is reversed using the LDEVTOVTAB procedure. Also, the *ext'vol* and *ext'ldev* can be the same array.

Copyright (C) 1989 Propellerhead Publishing

AWAKE (PCBPT, WAKECODE, WAITFLAGS)

PCBPT - offset in PCB table
(logical process table in
SYSGLOB)
1141

WAKECODE - events on which process
has to be awaken
(= WAKEMASK(in PCB) * CSR(Y))

WAITFLAGS -

<> ∅ then process goes
to sleep and
WAIT (WAITFLAGS, ∅)

WAIT (EVENTMASK, SPECIALINFO)

EVENTMASK - wait for events
(= EVENTFLAGS (in PCB) * CSR(Y)
if EVENTMASK < 0 then
return control immediatly
if an awake on one
of the event types has
already occured

SPECIALINFO -
(3:1) - enables
software
interrups

WAIT

PURPOSE

Used to block the calling process until the specified event occurs.

SYNTAX

```
                IV      LV
WAIT(eventmask, info)  O-P
```

PARAMETERS

eventmask (Integer by value, required)
This is the event that is to occur before the calling process is to continue executing. This bits in this mask are in the same order as those in the wakeup mask in the PCB table but are shifted right 4 bits. The following is a layout:

Bit	Event
5	Global Rin Wait
6	Local Rin Wait
7	Mail Wait
8	Blocked I/O Wait
9	I/O Wait
10	UCOP, RIT Wait
11	Junk Wait
12	Timer Wait
13	IPC Message Wait
14	Son Wait
15	Father Wait

info (Logical by value, required)
Contains information related to the cause of the wait, amount of time, or change in locality.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

Copyright (C) 1989 Propellerhead Publishing

DISCUSSION

Potential system failures include:

```
604 - Event has already occurred and the NEW wait
      procedure was used.
```


Copyright (C) 1989 Propellerhead Publishing

WHERES'DB

PURPOSE

Determines where the DB register is currently pointing.

SYNTAX

```
   L
dst = WHERES'DB  O-P
```

FUNCTION RETURN

Returns the DST number where the DB register is currently pointing. If DB is at an absolute memory address a value of %077777 will be returned.

PARAMETERS

None.

CONDITION CODES

CCL DB pointing to absolute memory address.
CCE DB is at the stack.
CCG DB pointing to a data segment other than the stack.

SPECIAL CONSIDERATIONS

Requires privileged mode.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

XCODETRAP

PURPOSE

Enables or disables code traps.

SYNTAX

```
                              IV        I
XCODETRAP(new'plabel,old'plabel)
```

PARAMETERS

new'plabel (Integer by value, required)
The address of the procedure to be invoked on a code trap. A number identifying the error that occurred is available to the trap procedure at Q-4. A zero value disables the trap.

old'plabel (Integer, required)
The address of the previous procedure designated to handle the trap.

CONDITION CODES

CCL Illegal plabel.
CCE Successful.
CCG Trap disabled.

SPECIAL CONSIDERATIONS

None.

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

XRETJTENTRY

PURPOSE

Returns a file equation entry from the JDT.

SYNTAX

```
 I                              BABA  BA  I      IA
flag = XRETJTENTRY(file, grp, acct, size, equation)  O-P
```

FUNCTIONAL RETURN

An flag indicating successful (0) or failure (1).

PARAMETERS

file (Byte array, required)
The formal file name.

grp (Byte array, required)
The formal group name.

acct (Byte array, required)
The formal account name.

size (Integer, required)
The maximum size of the equation array. This variable will be updated upon return with the actual length of the file equation.

equation (Integer array, required)
An array that will contain the returned file equation. See the System table manual for the file equation layout.

CONDITION CODES

Unchanged.

SPECIAL CONSIDERATIONS

Requires privileged mode.

Copyright (C) 1989 Propellerhead Publishing

DISCUSSION

None.

Copyright (C) 1989 Propellerhead Publishing

Section III - Expansions

Introduction

The following Expansions are extended discussions dealing with specific areas of the MPE-V low level procedures. A vast majority of the procedures documented in COMPASS are standalone. That is, they work by themselves, and perform a fairly isolated function. The descriptions and definitions in Section II will generally suffice to understand their use.

However, certain other of the MPE-V procedures documented in COMPASS fall into broad categories that encompass entire groups of procedures. Certain other procedures are sufficiently complex to warrant separate discussion. Foremost among these groups are the procedures with manipulate the directory, the security aspects of MPE-V, and the IO system, especially ATTACHIO. It is these areas that will be expanded in this section

Directory

Procedures: DIRECADJUST, DIRECFIND, DIRECFINDFILE, DIRECINSERT, DIRECINSERTFILE, DIRECLOGOFF, DIRECLOGON, DIRECPURGE, DIRECPURGEFILE, DIREC[RE]SETFLAG, DIRECSCAN, DIRMATCH

Probably the most commonly used set of procedures provided by MPE and documented in COMPASS are the directory procedures. These procedures allow the user to perform various functions upon the entities which reside in the directory and provide a common interface for locking and handling the various directory data structures.

Theory

MPE maintains a disc resident data structure called the directory. There is always at least one directory on a system, residing on the system disc. Private volume sets maintain their own directories, and these reside on the master volume of the volume set. Within the directory reside sub-structures, called entries, which catalog and describe various entities necessary for the bookkeeping and security functions of MPE. Specifically, these entities are Accounts, Groups within Accounts, Files within Groups, Users within Accounts, and Volume sets within groups.

The sub-structures within the directory contain data describing the entity and other data that is useful to maintain about the entity. For example, in an account entry, the account name appears, along with the account security, the account password, and connect time, CPU time, and disc space limits and current counters. These data generally can be set by MPE commands (such as :NEWACCT or :ALTACCT), or can appear in the printouts of various MPE commands (:REPORT or :LISTF for files). Other data that resides in the directory portion of MPE are pointers and index blocks that allow the physical and logical linkages to be maintained. Generally, these contain disc addresses that point to other structures or index blocks, allowing the various entities to be searched for and found relatively quickly. The formats of the individual

Copyright (C) 1989 Propellerhead Publishing

entries are documented in Section 4 of the MPE-V System Tables Manual, HP Part Number 32033-90119.

The directory procedures documented in COMPASS are used to traverse this directory structure, sort out the various pointers, and return or modify various entries in the directory.

Practice

There are four main functions that can be performed with the directory procedures: Finding, Purging, Inserting (adding), and Scanning. It can be argued that the last, Scanning, is a subset of Finding. As will be seen, however, scanning is complicated enough to warrant a separate discussion.[1]

The directory procedures have a common calling sequence, with a few exceptions. This means that most of the procedures in this group will be called with at least the first few parms the same as the others, and they return a common set of codes. In general, the calling sequence is:

```
status := DIRECproc (type, link,
                     acct, group, file....
   value type, link;
   integer type;
   double link;
   array acct, group, file;
```

One of the frustrating things about the directory procedures is that these standard parameters are not always treated exactly the same, have the same meaning, or are even used in every procedure. For example, there are several fields within the parameter *type* which only make sense for the procedure DIRECSCAN, as they are used for the function of scanning the directory. Further, the *type* parameter is really unnecessary for the procedure DIRECFINDFILE, as this procedure is designed to retrieve only file entries. In the case of these procedures, *type* is maintained for compatibility. Some procedures, such as DIRECADJUST, violate the standard and do **not** include these standard parms.

Another piece of data that MPE maintains are the disc addresses of certain key places in the directory. Specifically, when a user logs on, the disc address of the Account and Group index pointers (internal directory structures) that apply to that user are stored away in a session/job private data structure. These disc addresses can then be passed to the directory procedures in order to short cut the scanning process when locating directory entries in the user's logon group or account. This disc address is passed in the *link* parameter shown above. If it is necessary to scan the directory from the very highest level (called the Directory Root), then a double zero is passed in *link*, and the appropriate bits are set in the type parameter.

The *acct*, *group*, and *file* parameters also deserve some special attention. Not all three are necessary at all times. Their use varies with the value of *type*. For example, if DIRECFIND is being called with a *type* parameter indicating an account is being searched for, then *group* and *file* are ignored. To find a group entry, *file* is ignored, and so on. The *group* parameter can vary in function as well. It is really more properly called the *groupuser* parameter, as it can contain the user name if appropriate. If it is desired to scan for a specific user for example, then the user name must be passed in the *group* parameter.

---

[1]. Virtually all the directory procedures perform a scan of the directory to find the specific entry requested. This is necessary due to the underlying directory structure. The Scanning (with a capital S) referred to here is explicitly available to the user, and each entry encountered can be dealt with individually. Except where otherwise noted, scanning will refer to the internal, non-visible scan performed by the directory procedures.

Copyright (C) 1989 Propellerhead Publishing

Directory Expansion(continued)

File Entries

The directory routines handle all of the entries in the directory the same, except one, that being file entries. The reason for this is that certain operations on file entries also involve other levels of the directory. For example, building a file entails adding a file entry to the directory. The fact that a file is being built also implies that disc space (at least one extent's worth) is also being allocated, and therefore the group and account directory counts need to be modified. The same goes for when the file is purged. Because of this relationship of the file to the parent group and account, special procedures have been provided to adjust these disc space counters.

Another point to note about files is that the only part of the file that the directory procedures deal with the directory entry, and **not** the file label, data, or any other part of the file. The file entry is very simple, containing only the file's name (8 bytes), and disc address (4 bytes).

Private Volumes

MPE-V's implementation of removable media, called Private Volumes (PV) has a few idiosyncracies when dealing with directories. Each Volume Set has its own directory which resides on the Master Volume of the set. This directory contains a mirror image of the account and group entries which reside on the system volume (main directory) for that account on that PV set. User and volume set definition entries reside only in the system directory, and file entries reside only in the PV directory. Rather than hiding this separation of directories from the programmer, the directory procedures are set up such that the programmer must worry about which directory is being manipulated.

Some (unfortunately, not all) the directory procedures allow a parameter to be passed specifying the directory to be manipulated. This is usually called the *mvtabx* parameter, or Mounted Volume Table Index.[2] An *mvtabx* parameter of zero always indicates the system directory is to be manipulated. The *mvtabx* parm is returned by the procedure MOUNT (refer to page 2-193). Generally, the system directory is used to locate the group entry. This entry is then checked to see if the group resides on a Private Volume (the group's volume set name, group and account entries are not blank). If so, MOUNT is called to establish a link to the Volume Set indicated in the group's entry. The *mvtabx* returned by MOUNT is then passed to the appropriate directory procedure, which directs MPE to scan the PV's directory instead of the system. Usually, the caller would then perform a DISMOUNT (refer to page 2-100) to break the link to that PV set.

Examples

A couple of notes on the examples are in order. The examples which follow are, in general, working pieces of code. All were run before being included in this Expansion section. However, they have been shortened in the interest of space. Primarily, the error handling has been kept to a minimum. In actual production code, more error checking would need to be done. Specifically, a procedure to interpret and display the status from a failed directory call would be needed. The examples below show that only a QUIT is called when an error occurs. This is generally not good coding practice, as it gives no indication of what action to take.

Also, most of the programs are compiled with the $CONTROL PRIVILEGED option, which causes the programs to run in PM all the time. Depending upon individual coding style, you may wish to bracket the directory calls with GETPRIVMODE/GETUSERMODE calls.

---

[2]. This is the index into the Mounted Volume Table. There is an entry per mounted Private Volume.

Copyright (C) 1989 Propellerhead Publishing

Directory Expansion(continued)

Finding

The first and easiest function to perform with the directory routines is Finding. The routines used are for this purpose are DIRECFIND (page 2-69) and DIRECFINDFILE (page 2-72). DIRECFIND is used to locate any entry in the directory and return the raw buffer to the calling program. It is then up to the programmer to decipher and make sense of the data within the entry. The formats in the MPE-V Systems Tables manual become critical to being able to deal with the returns from these procedures.

The following SPL code sample shows a very simple example of how to retrieve an account entry. Refer to the definition of DIRECFIND on page 2-69 while examining the code. The program is very simple. It looks up the entry for the MATTEDOR account, then prints out the account password (words 10-13).

```
$CONTROL USLINIT,PRIVILEGED
BEGIN
  DOUBLE DRET, LINK;
  INTEGER LINK, MVTABX;
  ARRAY ACCT(0:4), GROUP(0:4), FILE(0:4), ENTRY(0:60);
  INTRINSIC GETPRIVMODE, GETUSERMODE, QUIT, PRINT;
  DOUBLE PROCEDURE DIRECFIND(TYPE, LINK, ACCT, GROUP, FILE, ENTRY);
    VALUE TYPE, LINK;
    INTEGER TYPE;
    DOUBLE LINK;
    ARRAY ACCT, GROUP, FILE, ENTRY;
    OPTION EXTERNAL;

BEGIN
   ACCT := "MATTEDOR ";     << SPACE FILL ARRAYS >>
   GROUP := 8(" ");
   FILE := 8(" ");
   TYPE := 0;         << ZERO OUT>>
   TYPE.(13:3) := 0;<< GLOBAL SEARCH >>
   TYPE.(10:3) := 2;<< ACCOUNT ENTRY >>
   LINK := 0D;        << SCANNING FROM DIRECTORY ROOT>>

   DRET := DIRECFIND(TYPE, LINK, ACCT,
                     GROUP, FILE, ENTRY);

   IF DRET <> 0D THEN QUIT(100); << SHOULD HAVE GOOD>>
                      << ERROR HANDLING>>
   PRINT(ENTRY(10),-8,0);
END;
END.
```

To find a group entry, the only changes would be the assignment of the *acct*, *group*, and *file* entries, plus the code loaded into *type*:

```
   ACCT := "MATTEDOR  ";     << SPACE FILL ARRAYS >>
   GROUP := "PUB      ";
   FILE := 8(" ");
   .
   .
   TYPE.(10:3) := 1;<<GROUP ENTRY>>
   .
   .
```


Copyright (C) 1989 Propellerhead Publishing

Directory Expansion(continued)

A file entry would be the same, the code in TYPE.(10:3) would be zero, and the file name loaded into FILE.

The procedure DIRECFINDFILE is used to find specific, file-related information. As can be seen by the definition of *freturn* on page 2-73, the data returned is the file's disc address, the file's parent group's security mask (as defined in the group entry) and the account security mask (if the TYPE.(13:3) is equal to 0 or 1). This is used by MPE to determine security access to the file for a specific user. The other field of the *type* parameter is ignored in the case of DIRECFINDFILE.

Calling DIRECFINDFILE is almost identical to DIRECFIND, except for the inclusion of the *mvtabx* parameter, and the return of *freturn*. The following code is a modified version of the DIRECFIND example above:

```
$CONTROL USLINIT,PRIVILEGED
BEGIN
  DOUBLE DRET, LINK;
  INTEGER TYPE, MVTABX;
  ARRAY ACCT(0:4), GROUP(0:4), FILE(0:4),          FRETURN(0:4);
  INTRINSIC GETPRIVMODE, GETUSERMODE, QUIT, PRINT;
  DOUBLE PROCEDURE DIRECFINDFILE(TYPE, LINK, ACCT,
                  GROUP, FILE, FRETURN,
      MVTABX);
    VALUE TYPE, LINK, MVTABX;
    INTEGER TYPE, MVTABX;
    DOUBLE LINK;
    ARRAY ACCT, GROUP, FILE, FRETURN;
    OPTION EXTERNAL;

BEGIN
   ACCT := "MATTEDOR  ";     << SPACE FILL ARRAYS >>
   GROUP := "PUB      ";
   FILE := "TEST1    ";
   TYPE := 0;         << ZERO OUT>>
   TYPE.(13:3) := 0;<< GLOBAL SEARCH >>
   TYPE.(10:3) := 2;<< ACCOUNT ENTRY >>
   LINK := 0D;        << SCANNING FROM DIRECTORY ROOT>>
   MVTABX := 0;       << SYSTEM DIRECTORY>>

   DRET := DIRECFINDFILE(TYPE, LINK, ACCT,
                         GROUP, FILE, FRETURN,
                         MVTABX);

   IF DRET <> 0D THEN QUIT(100); << SHOULD HAVE GOOD>>
                                 << ERROR HANDLING>>
   PRINT(ENTRY(10),-8,0);
END;
END.
```

Purging

The next most common application of the directory procedures is to purge, or remove, entries from the directory. As with any modification of an MPE system table, **extreme care must be used.** Fortunately, the

Copyright (C) 1989 Propellerhead Publishing

directory purge routines DIRECPURGE and DIRECPURGEFILE are fairly robust. When one level of the directory is removed (such as an account), the levels below it are automatically removed. Again, these procedures are very straightforward to call. The following code would purge the MATTEDOR account that we found in the example code above:

```
$CONTROL USLINIT,PRIVILEGED
BEGIN
  DOUBLE DRET, LINK;
  INTEGER LINK, MVTABX;
  ARRAY ACCT(0:4), GROUP(0:4), FILE(0:4), ENTRY(0:60);
  INTRINSIC GETPRIVMODE, GETUSERMODE, QUIT, PRINT;
  DOUBLE PROCEDURE DIRECPURGE(TYPE, LINK, ACCT, GROUP, FILE,
MVTABX);
    VALUE TYPE, LINK, MVTABX;
    INTEGER TYPE, MVTABX;
    DOUBLE LINK;
    ARRAY ACCT, GROUP, FILE;
    OPTION EXTERNAL;

BEGIN
   ACCT := "MATTEDOR  ";     << SPACE FILL ARRAYS >>
   GROUP := 8(" ");
   FILE := 8(" ");
   TYPE := 0;          << ZERO OUT>>
   TYPE.(13:3) := 0;<< GLOBAL SEARCH >>
   TYPE.(10:3) := 2;<< ACCOUNT ENTRY >>
   LINK := 0D;         << LOCATE FROM DIRECTORY ROOT>>
   MVTABX := 0;        << SYSTEM DIRECTORY>>

   DRET := DIRECPURGE(TYPE, LINK, ACCT,
                      GROUP, FILE, MVTABX);

   IF DRET <> 0D THEN QUIT(100); << SHOULD HAVE GOOD>>
                                 << ERROR HANDLING>>
END;
END.
```

The same method can be used to remove a user, group, or volume set definition entry. Again, all the lower levels will be removed when a higher level is removed. In the example above, all group, user and file entries were removed by the one call to DIRECPURGE. Additionally, the disc space occupied by the files was returned to free space.

As with searching for file entries, removing file entries can be done in one of two ways. If DIRECPURGE is called and the code for file entries is passed (`TYPE.(10:3) = 0`), then only the directory entry is removed, **leaving the file label and all allocated space as is.** This is useful when a file label is damaged and it is desirable to restore a known good version of the file. The utility FLUTIL in the TELESUP account can purge files in this manner if necessary. Using DIRECPURGE leaves the file space allocated and thus the account and group file space counts incorrect.[3]

---

[3]. The file space can be recovered by doing a recover lost disc space during a coolstart. The file space counters can be repaired by manually changing them, perhaps with DIRECADJUST. A safer way is to store the files, purge the groups or accounts affected, rebuild them, and restore the files with OLDDATE. The counts are updated correctly by RESTORE as the files are brought onto the system.

Copyright (C) 1989 Propellerhead Publishing

Directory Expansion(continued)

The correct way to remove a file and its associated disc space is to use DIRECPURGEFILE. This intrinsic differs from DIRECPURGE in that it accepts a *numsects* parameter, which tells the directory system by what amount to adjust the group and account disc space counters. Negative amounts indicate the counts are to be decremented, positive indicate they are to be incremented.

The easiest way to determine the amount of disc space to be decremented when DIRECPURGEFILE is used is by using the FSECTORS procedure (see page 2-128). This procedure accepts a file label as input, and returns the number of sectors occupied by the file. The following code sample shows how to use DIRECFINDFILE, FLABIO, FSECTORS and DIRECPURGEFILE to purge a file and keep the sector counts in the group and account entries correct:

```
$CONTROL USLINIT,PRIVILEGED,MAP
BEGIN
  ARRAY          FRETURN     (0:4),
                 ACCT        (0:4),
                 GROUP       (0:4),
                 FILE        (0:4),
                 FLABEL      (0:127);

  DOUBLE ARRAY D'FRETURN(*) = FRETURN;

  DEFINE FRET'VTABX          = FRETURN(0).(0:8)#,
         FRET'DISC'ADDR      = D'FRETURN#;

  DOUBLE RET,
         LINK,
         SECTOR'ADJUST;

  INTEGER LDEV,
          MVTABX,
          FLABRET,
          TYPE;

  INTRINSIC DEBUG,QUIT;
  DOUBLE PROCEDURE DIRECFINDFILE(TYPE,LINK,ACCT, GROUP,
                          FILE,  FRETURN,  MVTABX);
     VALUE TYPE, LINK, MVTABX;
     INTEGER TYPE, MVTABX;
     DOUBLE LINK;
     ARRAY ACCT, GROUP, FILE, FRETURN;
     OPTION EXTERNAL, VARIABLE;

  INTEGER PROCEDURE LUN(VTABX, MVTABX);
     VALUE VTABX, MVTABX;
     INTEGER VTABX, MVTABX;
     OPTION EXTERNAL;

  INTEGER PROCEDURE FLABIO(LDEV, ADDRESS, FUNC, FLABEL);
     VALUE LDEV, ADDRESS, FUNC;
     INTEGER LDEV, FUNC;
     DOUBLE ADDRESS;
     ARRAY FLABEL;
     OPTION EXTERNAL;
```

Copyright (C) 1989 Propellerhead Publishing

Directory Expansion(continued)

```
  DOUBLE PROCEDURE FSECTORS(FLABEL);
     ARRAY FLABEL;
     OPTION EXTERNAL;

  DOUBLE PROCEDURE DIRECPURGEFILE(NUMSECTS, DUMMY, ACCT,
                                  GROUP, FILE, MVTABX);
     VALUE NUMSECTS, DUMMY, MVTABX;
     INTEGER DUMMY, MVTABX;
     DOUBLE NUMSECTS;
     ARRAY ACCT, GROUP, FILE;
     OPTION EXTERNAL, VARIABLE;

BEGIN
   MOVE FILE := "TESTFILE ";
   MOVE GROUP := "JASON   ";
   MOVE ACCT := "MATTEDOR ";
   TYPE.(13:3) := 0; << GLOBAL SEARCH >>
   TYPE.(10:3) := 0; << FILES >>
   LINK := 0D;       << GLOBAL >>
   MVTABX := 0;      << SYSTEM DIRECTORY>>
   RET := DIRECFINDFILE(TYPE,LINK, ACCT, GROUP, FILE,
                        FRETURN, MVTABX);
   IF RET <> 0D THEN QUIT(100);

   LDEV := LUN(FRET'VTABX,0); << SYSTEM DIRECTORY>>

   FRET'VTABX := 0; << ZERO TO MAKE ADDRESS CORRECT>>

<< FUNCTION CODE OF ZERO IS READ >>
   FLABRET := FLABIO(LDEV, FRET'DISC'ADDR, 0, FLABEL);
   IF FLABRET <> 0 THEN QUIT(1000+FLABRET);

   SECTOR'ADJUST := FSECTORS(FLABEL);

   RET := DIRECPURGEFILE(-SECTOR'ADJUST, 0, ACCT, GROUP,
                FILE, 0);
   IF RET <> 0D THEN QUIT(200);
END;
END.
```

Inserting

As with most aspects of the internals of MPE-V, it is rarely wise to modify any of its structures. This especially applies to the directory. Inserting objects into the directory is definitely not for the faint of heart. As opposed to inserting an entry in a normal MPE memory table, the directory is more complex because most directory entries have relationships to other entries. For example, to create an account, one must create the account entry, a user entry (for the account manager) and a PUB group entry. (In this case, it is infinitely wiser to just issue a programmatic :NEWACCT command and let MPE do the work).

All this aside, there are legitimate times when a directory entry should be inserted into the directory. Such utilities as backup programs (STORE, BACKUP, BACKPACK) must be able to create users, groups and accounts. Further it must insert the file entries into the directory when a file is "created" on the restore.

Copyright (C) 1989 Propellerhead Publishing

Directory Expansion(continued)

The procedure DIRECINSERT has a bizarre idiosyncracy which none of the other procedures share. Instead of passing the entire formatted entry to be added to the directory, the entry passed to DIRECINSERT is everything **except** the first four words, or the name. In other words, the entry passed starts at the group index pointer (in the case of account entries) or the file index pointer (in the case of group entries). The name of the entry is taken from the *group* or *account* name passed explicitly in the parms. This name is inserted by DIRECINSERT in front of whatever is passed in *ntry* to form the complete directory entity.

The example shown below is fairly simple, but it illustrates the necessary principles for directory entry insertion. The hardest part is getting the directory entry formatted correctly. Note that what is actually passed to DIRECINSERT is GROUP'ENTRY(4), the file index pointer. The entire group entry is defined and formatted in this example for clarity. The code to insert a group into an existing account is:

```
$CONTROL USLINIT,PRIVILEGED
BEGIN
   ARRAY              GROUP'ENTRY          (0:40);
   BYTE ARRAY         B'GROUP'ENTRY        (*) = GROUP'ENTRY;

   DOUBLE POINTER     DP             := @GROUP'ENTRY(9);

   DEFINE    GNAME                    = B'GROUP'ENTRY#,
             GFIPNTR                  = GROUP'ENTRY(4)#,
             GPASS                    = B'GROUP'ENTRY(10)#,
             GDFSCOUNT                = DP#,
             GDFSLIMIT                = DP(1)#,
             GCPUCOUNT                = DP(2)#,
             GCPULIMIT                = DP(3)#,
             GCONTIMECOUNT            = DP(4)#,
             GCONTIMELIMIT            = DP(5)#,
             GSEC1                    = GROUP'ENTRY(21)#,
             GSEC2                    = GROUP'ENTRY(22)#,
             GCAPABILITY              = GROUP'ENTRY(23)#,
             GLINKAGE                 = GROUP'ENTRY(24)#,
             GVSDIPNTR                = GROUP'ENTRY(25)#,
             GHVSNAME                 = B'GROUP'ENTRY(52)#,
             GHVSGNAME                = B'GROUP'ENTRY(60)#,
             GHVSVSNAME               = B'GROUP'ENTRY(68)#,
             GSAVEFIPNTR              = GROUP'ENTRY(38)#,
             GMOUNTREFCNTR            = GROUP'ENTRY(39)#,
             GSPARE                   = GROUP'ENTRY(40)#;

ARRAY                ACCT                 (0:4),
                     GROUP                (0:4),
                     FILE                 (0:4);

INTEGER              TYPE,
                     MVTABX;

DOUBLE               DRET,
                     LINK;

INTRINSIC            QUIT,
                     DEBUG;
```

Copyright (C) 1989 Propellerhead Publishing

Directory Expansion(continued)

```
DOUBLE PROCEDURE DIRECINSERT (TYPE, LINK, ACCT,
                              GROUP, FILE, NTRY, MVTABX);
  VALUE TYPE, LINK, MVTABX;
  INTEGER TYPE, MVTABX;
  DOUBLE LINK;
  ARRAY ACCT, GROUP, FILE, NTRY;
  OPTION EXTERNAL, VARIABLE;

BEGIN
  MOVE GNAME := "TESTGRP ";
  GFIPNTR := 0;
  MOVE GPASS := "TESTPASS";
  GDFSCOUNT := GCPUCOUNT := GCONTIMECOUNT := 0D;
  GDFSLIMIT := GCPULIMIT := GCONTIMELIMIT :=
       %17777777777D;

<< SET ACCESS AT R,W,A,L,X,S:AC,AL>>
  GSEC1 := %(2)0001100011000110;
  GSEC2 := %(2)0011000110001100;
  GCAPABILITY := %000600;       << IA, BA >>

  GLINKAGE := GVSDIPNTR := 0;
  MOVE GHVSNAME := 8(" ");
  MOVE GHVSGNAME := 8(" ");
  MOVE GHVSVSNAME := 8(" ");
  GSAVEFIPNTR := GMOUNTREFCNTR := GSPARE := 0;

  TYPE.(13:3) := 0;  <<GLOBAL>>
  TYPE.(10:3) := 1;  <<GROUP>>

  LINK := 0D;
  MVTABX := 0;  << SYSTEM DIR>>

  MOVE ACCT := "MATTEDOR";
  MOVE GROUP := "TESTGRP ";
  MOVE FILE := 8(" ");

  DRET := DIRECINSERT(TYPE, LINK, ACCT, GROUP,
                      FILE, GROUP'ENTRY(4), MVTABX);
  IF DRET <> 0D THEN QUIT(200);

END;
END.
```

An example will not be included, but insertion of file entries is done with DIRECINSERTFILE (page 2-78), just as purging was done with DIRECPURGEFILE. Just as with purging, DIRECINSERTFILE accepts the *numsects* parm to adjust the group and account file space counts. In this case, *numsects* should be a positive number, which will increment the counters. No entry is passed, but the *file*, *group*, and *acct* define the file name, and the *faddr* parm defines the file address. The file name and address constitute the entire file entry in the directory, so in effect, the entry is being passed. The only other idiosyncracy of DIRECINSERTFILE is that the *dummy* parameter is passed, an integer by value. This makes the number of words of parameters passed in the first 2 parameters equal to the a normal call (*type* and *link* at 3 words each, *numsects* and *dummy* at 3 words each).

Copyright (C) 1989 Propellerhead Publishing

Directory Expansion(continued)

Scanning

By far the most commonly used and easily the most obtuse function of the directory system in MPE-V is Scanning, implemented by the procedure DIRECSCAN (page 2-90).

DIRECSCAN does not operate the way you might expect. In any scanning operation of a tree structure like the directory, it is necessary for the procedure doing the scanning to "remember" where it was in the tree structure. It must also pass control to the user's code to deal with each entry visited so that the application can perform the tasks it must perform. Under normal circumstances, the procedure doing the scanning would "forget" (its local storage would be deallocated) where it was.

Thus, the problem presented to the designers of the directory scanning procedure was to be able to scan the directory efficiently and remember where to resume the scan after the user's code executed. These two key requirements can be accomplished by one of two basic methods. The first is to have the caller (user) provide a workarea so that DIRECSCAN can save whatever data it needs from one call to the next. In this manner, DIRECSCAN would be called and would return a directory entry that the calling program (user code) could process as the application saw fit. DIRECSCAN would also stuff some data away in an area the caller provided, but did not touch. Then DIRECSCAN would be called again to return the next entry.

While this method is very straightforward, the method DIRECSCAN's authors chose is a bit less intuitive. In the method used, a procedure is provided by the user to deal with each entry. DIRECSCAN is called only once, and one of the parameters passed is the plabel (code address) of the user procedure (commonly called the RECIP (reciprocating) procedure). This procedure is then called **within** DIRECSCAN whenever a directory entry is visited, with the directory entry being passed to this procedure as a parameter. Thus the situation in the two methods are the reverse of each other. In one, the user's code is the driver, and the directory procedure is the called module. In the method actually employed in MPE-V, however, the directory procedure (DIRECSCAN) is the driver, with the user procedure (RECIP) the called module. This method relies heavily on the underlying HP 3000 architecture, as we will see.

DIRECSCAN Parms

For the most part, the parameters passed to DIRECSCAN follow the generic format mentioned at the beginning of this section: a *type* parameter, *link*, etc. However, there are a couple of extra parms that must be discussed, and the *type* parameter has a few extra fields that are crucial to making DIRECSCAN work properly. Refer to the definition of DIRECSCAN on page 2-90.

The *type* parameter has some of the same fields as the *type* parameter of the other procedures. `TYPE.(13:3)` still specifies what type of search to perform, global or using the *link* parameter to search the logon account or group. `TYPE.(10:3)` still specifies what type of entry to scan for, but this now has a slightly different meaning. This field is called the ENDLEVEL, and specifies what level of the directory the *acct*, *group*, and *file* entries specify. This is NOT the level of the directory that we ultimately wish to scan, visit and process. That is specified in the TOLEVEL subfield of *type* in `TYPE.(6:3)`. The values for both fields are the same: 0 = file, 1 = group, 2 = account, and so on.

At first it may seem that there is one too many fields specifying entry types exist in the *type* parameter. This may be true, but the extra parameters give us more flexibility, and we can use them to shortcut the scan if we wish. The key thing to remember is that the names passed in *acct*, *group*, and *file* parameters specify what ENDLEVEL entities are to be visited. The entries we wish to visit, however, are specified in the TOLEVEL field. Thus, if the *acct* parameter contains the name "GOERTZ" and the ENDLEVEL field contains a 2 (account), this tells DIRECSCAN to search for and find the GOERTZ account entry. If the TOLEVEL field contains a zero (files), then DIRECSCAN, once it finds the GOERTZ account entry, will scan down each group and allow our RECIP procedure to examine each file entry. Thus, the fields together

Copyright (C) 1989 Propellerhead Publishing

can be read to mean "Start at STARTLEVEL, then find the entities specified by ENDLEVEL. Once there, visit the TOLEVEL entries". But wait, there's more.

`TYPE.(9:1)` is usually called the ALLFLAG, and specifies that all of whatever ENDLEVEL has been specified should be visited. This is sometimes a handy shorthand method of specifying that a global search is to be done. For example, if we want to scan all files in the directory, all have to do is specify an ENDLEVEL of 2 (Account), set the ALLFLAG on (visit all accounts) and set TOLEVEL to zero (files). In this case, the *acct*, *group*, and *file* entries do not even have to be set.

One last field of the *type* parameter to be mentioned is the HITFLAG, or `TYPE.(5:1)`. This specifies that each entry visited on the way to TOLEVEL is to be returned to the RECIP procedure for processing. In some cases, this will occur anyway, and the RECIP procedure should be able to deal with these entries no matter what. A good example of this is if a STARTLEVEL of 0 is specified (global search), the TOLEVEL and ENDLEVEL are both set to files, and the HITFLAG is on. This would normally scan whatever group and account were passed in, pass each file entry to the RECIP procedure, then terminate. Only file entries would be passed to the RECIP procedure in this scenario. However, if the HITFLAG were turned on, the account entry and group entry visited on the way to the files would **also** be passed to the RECIP procedure. The HITFLAG is really only marginally useful, however. Normally, if the TOLEVEL and ENDLEVEL are different (especially if a STARTLEVEL of zero is specified), all the levels between TOLEVEL and ENDLEVEL, inclusive, will be returned to DIRECSCAN whether the HITFLAG is turned on or not.

Fortunately, most of the time we do not have to fool with the complexities of the *type* parameter too much. The procedure PRODUCEPARMS (page 2-207) will accept as input a file/group/account set (such as would be entered on a :LISTF or :REPORT command) along with the desired TOLEVEL (what entities in the directory we are really interested in), parse the input, and return all the parameters needed for a good call to DIRECSCAN (except *mvtabx* and *recip*). The example program in this section shows the use of PRODUCEPARMS.

The RECIP Procedure

The next level of complexity in the scanning operation comes in defining the RECIP procedure. As mentioned, this procedure automatically called by DIRECSCAN whenever a directory entry is visited that meets the criteria specified in DIRECSCAN's parms.

Since DIRECSCAN must be able to call any generic procedure passed, the procedure that is passed to it must conform strictly to DIRECSCAN's assumptions about the procedure. This boils down to the calling sequence (parameters) of the RECIP procedure, and any information that RECIP returns to DIRECSCAN.

RECIP is defined as follows:

```
INTEGER PROCEDURE RECIP (NTRY, LEVEL, PARMS, SIRS);
  VALUE LEVEL, PARMS, SIRS;
  INTEGER LEVEL, PARMS;
  DOUBLE SIRS;
  ARRAY NTRY;
```

The name, of course, does not have to be RECIP and can be anything, as long as that same name is passed in the *recip* parameter of the DIRECSCAN call. Before discussing the parameters, it is necessary to understand the environment that DIRESCSAN sets up before calling RECIP.

When RECIP is entered, DIRECSCAN has done the following: 1. SETCRITICAL. 2. Called EXCHANGEDB to point the DB register to the base of the Directory Data Segment (DST 20(10)), and 3. Locked the Directory SIR. The reason for this was that it was felt by DIRECSCAN's authors that the

Copyright (C) 1989 Propellerhead Publishing

RECIP procedure could be made very fast if it did not have to do these things. However, the program is not very safe. Aborting the program with any one of these conditions set will crash the system. In the example below, all are reset at the beginning of RECIP and then re-instated at the end. While this is slow (EXCHANGEDB, SETCRITICAL/RESETCRITICAL and GETSIR/RELSIR all call ERRORON, a fairly high overhead MPE procedure), it makes the code infinitely safer. The window of dangerous code is just a few lines at either end of the RECIP procedure rather than the whole thing.

It is necessary to deal with the split-stack mode that the procedure is in when entered. DIRECSCAN uses the Directory Data Segment for its data buffer when reading the directory. The NTRY parameter of RECIP is really a DDS relative address of where the entry can be found. In order to use this safely, RECIP must copy the data to the calling stack. The example below does this by using the MVBL (MoVe dB to dL) instruction, one of only two move instructions which operate in split stack mode. This instruction copies data from a DB relative address to a DL relative address. To make things easy, it just copies the entry to beginning of the DL area in the stack (word 0). Once the stack is re-acquired (EXCHANGEDB(0)), a pointer is set to the DL area, and the entry treated like a normal array of data.

LEVEL is just a code that indicates what level of the directory NTRY belongs to. These values are the same as TOLEVEL and ENDLEVEL. SIRS is a double word that contains the SIR number and the flag from GETSIR indicating whether the SIR had already been acquired. These are usually just stacked and RELSIR called indicating that the stacked parameters are to be used.

One other parameter that is passed to RECIP is the PARMS parameter. This is the same PARMS array passed in the *parms* parameter of DIRECSCAN. The idea behind PARMS is simple. It is always necessary to pass some specific data to RECIP, for example the generic file set that we want to scan (RECIP will see all entries, and will have to select the ones to process. More on this later). Since the RECIP interface was designed to run in split-stack mode for speed, and to run as an external procedure not necessarily in the same compilation unit as the caller, this interface was necessary.

Notice, however, that the parameter is not defined as an array as it is for DIRECSCAN, but an Integer by Value. What is actually passed to RECIP is an address, relative to DIRECSCAN's Q register of the PARMS array passed to DIRECSCAN. Care has to be taken in split-stack mode not to generate DB relative address, thus everything is Q relative. To set the pointer, we first find our own Q, calculate the number of words back to the previous (DIRECSCAN's) Q, then add the address passed in PARM to RECIP. Note this will usually be negative, as it must reflect the address of the original array passed in the outer call to DIRECSCAN.

Since we will not be running in split-stack mode in RECIP, we do not have to take quite as much care. The example below sets a pointer to parms, but since the RECIP is part of the same compilation unit as the mainline, it has access to all the globally defined data areas. These are used directly. A pointer is set to PARMS for illustration purposes, but is not used in the procedure.

A Working Program

The following example is a program called SCANDEMO. It was originally published in the HP Chronicle, and the version presented here has been revised a bit:

```
$CONTROL USLINIT,PRIVILEGED,MAP
$TITLE  " PROGRAM SCANDEMO --- DECLARATIONS"
BEGIN
  <<     PROGRAM:      SCANDEMO

PURPOSE:  THIS PROGRAM SERVES AS A DEMONSTRATION ON HOW TO USE THE
          MPE PROCEDURE DIRECSCAN.
```

Copyright (C) 1989 Propellerhead Publishing

Directory Expansion(continued)

```
AUTHOR:    JASON M. GOERTZ, MATTEDOR COMPUTER SERVICES

SUMMARY:   THIS PROGRAM READS A FILE SET FROM THE      USER, THEN
           CALLS PRODUCEPARMS AND DIRECSCAN TO SCAN THE DIRECTORY
           FOR THAT SET.  AT EACH ENTRY, A RECORD IS DISPLAYED WITH
           THE ENTRY TYPE, (IE, FILE, GROUP, ETC) AND CERTAIN PIECES
           OF INFO ABOUT THE ITEM.   THE FUNCTIONALITY IS MINIMIAL,
           BUT IT IS INTENDED PRIMARILY TO BE MODIFIED BY THE USER
           FOR A SPECIFIC PURPOSE

COPYRIGHT 1989, MATTEDOR COMPUTER SERVICES, SUMNER, WASHINGTON ALL
RIGHTS RESERVED.  THIS PROGRAM MAY BE USED AND MODIFIED AS DESIRED,
AS LONG AS COPYRIGHT BANNER IS NOT MODIFIED

           COMPILATION INSTRUCTIONS:

:SPL SCANDEMO,,$NULL
:PREP $OLDPASS,SCANP;MAXDATA=20000;&
CAP=IA,BA,PM;STACK=2000
:SAVE SCANP

>>

<< DEFINITIONS FOR DISPLAY STATEMENTS >>

DEFINE            DISPLAY    = BEGIN MOVE BBUFF := #,
                  ENDDISP    = ,2; LEN := TOS - @BBUFF;
                  PRINT(LBUFF,-LEN,0); END#,
                  NOCRLF     = ,2; LEN := TOS - @BBUFF;
                PRINT(LBUFF,-LEN,%320);END#;
DEFINE            CLEARBUFF = MOVE BBUFF := " ",2;
                         MOVE * := BBUFF,(BUFFLEN-1)#;

EQUATE            BUFFLEN              = 80;
ARRAY             LBUFF             (0:BUFFLEN/2);
BYTE ARRAY        BBUFF             (*) = LBUFF;
INTEGER           LEN;
LOGICAL           PARM                 = Q-4;

<<  DEFINITIONS FOR PRODUCEPARMS/DIRECSCAN  >>

EQUATE            FILE'LEVEL           = 0,
                  GROUP'LEVEL          = 1,
                  ACCOUNT'LEVEL        = 2,
                  USER'LEVEL           = 3,
                  VOLUME'LEVEL         = 4;

ARRAY             LA'RESULT         (0:30);
BYTE ARRAY        BA'RESULT         (*) = LA'RESULT;
DOUBLE ARRAY      DA'RESULT         (*) = LA'RESULT;

DEFINE            PP'INDEX               = DA'RESULT#,
     PP'TYPE                  = LA'RESULT(2)#,
```


Copyright (C) 1989 Propellerhead Publishing

Directory Expansion(continued)

```
PP'D'BA'FILENAME       = BA'RESULT(6)#,
PP'D'LA'FILENAME     = LA'RESULT(3)#,
                  PP'D'BA'GROUPNAME    = BA'RESULT(14)#,
                  PP'D'LA'GROUPNAME    = LA'RESULT(7)#,
                  PP'D'BA'ACCTNAME     = BA'RESULT(22)#,
                  PP'D'LA'ACCTNAME     = LA'RESULT(11)#,
                  PP'D'BA'LOCKWORD     = BA'RESULT(30)#,
                  PP'D'LA'LOCKWORD     = LA'RESULT(15)#,
                  PP'G'BA'FILENAME     = BA'RESULT(38)#,
       PP'G'LA'FILENAME     = LA'RESULT(19)#,
                  PP'G'BA'GROUPNAME    = BA'RESULT(46)#,
                  PP'G'LA'GROUPNAME    = LA'RESULT(23)#,
                  PP'G'BA'ACCTNAME     = BA'RESULT(54)#,
                  PP'G'LA'ACCTNAME     = LA'RESULT(27)#;

INTEGER           ERROR, << CI ERROR RETURNED BY PP >>
                  I;

LOGICAL           FLAG;

BYTE ARRAY        DELIM                (0:10);

DOUBLE            DRET;
INTEGER           DRET1                = DRET,
                  DRET2                = DRET1 + 1;

<<   VARIABLE FOR PROCESS'AN'ENTRY.   NORMALLY, THESE WOULD BE
PROCEDURE LOCAL VARIABLES.  SINCE WE ENTER IN SPLIT STACK MODE, THE
ARRAYS CAN'T BE SET UP CORRECTLY.
>>
INTEGER           DDSDST;

POINTER           DL'POINT'LA;

BYTE POINTER      DL'POINT'BA;

DOUBLE POINTER    DL'POINT'DA;

BYTE ARRAY        REALNAME           (0:8),
                  DESIGNATOR         (0:8);

<< PROCEDURE DECLARATIONS >>

DOUBLE PROCEDURE DIRECSCAN (TYPE,LINK,ACCT,GROUP,FILE,
                            PROC,PARMS,VOLUME);
  VALUE TYPE,LINK,VOLUME;
  INTEGER TYPE,VOLUME;
  DOUBLE LINK;
  ARRAY ACCT,GROUP,FILE,PARMS;
  INTEGER PROCEDURE PROC;
  OPTION EXTERNAL,VARIABLE,PRIVILEGED;

LOGICAL PROCEDURE PRODUCEPARMS (LEAF,NAME,RESULT,DELIM,ERROR);
  VALUE LEAF;
```

Copyright (C) 1989 Propellerhead Publishing

Directory Expansion(continued)

```
  INTEGER LEAF,ERROR;
  BYTE ARRAY NAME,DELIM;
  LOGICAL ARRAY RESULT;
  OPTION EXTERNAL,PRIVILEGED;

INTEGER PROCEDURE DIRMATCH(DESIGNATOR,REALNAME);
  VALUE DESIGNATOR,REALNAME;
  BYTE POINTER DESIGNATOR,REALNAME;
  OPTION EXTERNAL;

LOGICAL PROCEDURE EXCHANGEDB(DST);
  VALUE DST;
  LOGICAL DST;
  OPTION EXTERNAL;

LOGICAL PROCEDURE SETCRITICAL;
  OPTION EXTERNAL;

PROCEDURE RESETCRITICAL(I);
  VALUE I;
  INTEGER I;
  OPTION EXTERNAL;

PROCEDURE GETSIR(SIR);
  VALUE SIR;
  LOGICAL SIR;
  OPTION EXTERNAL;

PROCEDURE RELSIR(SIR,FLAG);
  VALUE SIR,FLAG;
  LOGICAL SIR,FLAG;
  OPTION EXTERNAL;

INTEGER PROCEDURE PROCESS'AN'ENTRY(ENTRIE,LEVEL,PARMS,SIR);
  VALUE LEVEL,PARMS,SIR;
  INTEGER LEVEL,PARMS;
  DOUBLE SIR;
  ARRAY ENTRIE;
  OPTION FORWARD;  << FORWARD DECLARATION OF RECIP PROCEDURE >>

PROCEDURE GENMSGU(SETNO,MSG);
  VALUE SETNO,MSG;
  INTEGER SETNO,MSG;
  OPTION EXTERNAL;

INTRINSIC        DEBUG,
                 GETUSERMODE,
                 GETPRIVMODE,
                 PRINT,
                 READX,
                 ASCII,
                 DASCII,
                 BINARY,
                 QUIT,
```


Copyright (C) 1989 Propellerhead Publishing

Directory Expansion(continued)

```
                DBINARY;

<<    PROCEDURE:    PROCESS'AN'ENTRY

      THIS PROCEDURE PROCESSES EACH ENTRY PASSED TO IT AND IS CALLED
      AS THE RECIP PARAMETER OF DIRECSCAN.  THIS PROCEDURE IS THE
      MAIN ONE THAT CAN BE MODIFIED TO CHANGE THE FUNCTION OF THIS
      PROGRAM
>>
INTEGER PROCEDURE PROCESS'AN'ENTRY(ENTRIE,LEVEL,PARMS,SIRS);
  VALUE LEVEL,PARMS,SIRS;
  INTEGER LEVEL,PARMS;
  DOUBLE SIRS;
  ARRAY ENTRIE;

<<    UPON ENTRY TO THIS PROCEDURE, THE DB REGISTER IS SET TO THE
DIRECTORY DATA SEGMENT, THE PROCESS IS SETCRITICAL, AND THE
DIRECTORY SIR IS LOCKED.  THE FIRST PART OF THIS PROCEDURE UNDOES
ALL OF THIS, SO THAT WE CAN WORK FREELY WITHOUT FEAR OF RETRIBUTION
FROM MPE.

THE FIRST THING WE HAVE TO DO IS MOVE THE DATA FROM THE DDS TO OUR
STACK.  THE EASIEST WAY IS TO DO AN MVBL INSTRUCTION, WHICH MOVES
DATA DB RELATIVE TO DL RELATIVE, ALLOWING SPLIT STACK MOVES
>>
BEGIN

INTEGER RECIP'RET;
INTEGER DELTAQ = Q-0;
POINTER PARMS'PNT;

   TOS := 0;          << DL RELATIVE >>
   TOS := @ENTRIE;    << DB RELATIVE >>
   TOS := 29;         << LENGTH OF LONGEST ENTRY >>
   ASSEMBLE (MVBL);

<<  NEXT, EXCHANGE BACK TO OUR STACK, SAVING THE DST NUMBER FOR THE
DDS SO WE CAN EXCHANGE BACK AT THE END OF THE PROCEDURE
>>
    DDSDST := EXCHANGEDB(0);    << 0 = OUR STACK >>

<<  NEXT, RELEASE THE SIR >>

    TOS := SIRS;
    RELSIR(*,*);

<<  NOW, RESET THE CRITICAL BIT >>

    RESETCRITICAL(0);
    GETUSERMODE;

<<  NOW WE CAN BREATH EASY.  SET A POINTER IN THE DL AREA TO ALLOW
ACCESS TO THE DATA.  SEE TABLES MANUAL FOR ENTRY FORMATS
>>
```

Copyright (C) 1989 Propellerhead Publishing

Directory Expansion(continued)

```
    RECIP'RET := 0;
    RECIP'RET.(15:1) := 0;  << RELEASED THE SIRS >>

    PUSH(DL);
    @DL'POINT'LA := TOS;
    @DL'POINT'BA := @DL'POINT'LA & ASL(1); << SET BYTE
                                  POINTER >>
    @DL'POINT'DA := @DL'POINT'LA;        << SET LOGICAL
                                  POINTER >>

<< SET POINTER TO PARMS ARRAY >>

   PUSH(Q);
   @PARMS'PNT := TOS; !GET OUR Q
   @PARMS'PNT := @PARMS'PNT - DELTAQ; !POINT TO
                                        DIRECSCAN'S Q
   @PARMS'PNT := @PARMS'PNT + PARMS; !ADD @ OF PARMS

<<  CALL DIRMATCH TO MAKE SURE THEY MATCH
>>
    MOVE REALNAME := 9(" ");
    MOVE DESIGNATOR := 9(" ");
    MOVE REALNAME := DL'POINT'BA,(8);
    IF  LEVEL = FILE'LEVEL THEN
        MOVE DESIGNATOR := PP'G'BA'FILENAME,(8)
    ELSE IF LEVEL = GROUP'LEVEL THEN
        MOVE DESIGNATOR := PP'G'BA'GROUPNAME,(8)
    ELSE IF LEVEL = ACCOUNT'LEVEL THEN
        MOVE DESIGNATOR := PP'G'BA'ACCTNAME,(8);

    GETPRIVMODE;
    I := DIRMATCH(DESIGNATOR,REALNAME);
    IF  I > 0 THEN     ! DESIG GREATER THAN REAL.
                       ! WANT TO SKIP BUT KEEP
     BEGIN             ! SCANNING.
      IF LEVEL = GROUP'LEVEL OR LEVEL = ACCOUNT'LEVEL
THEN
         RECIP'RET.(13:2) := 1 << SKIP THIS SUBTREE >>
      ELSE IF LEVEL = FILE'LEVEL THEN ;
      GO TO GET'LOST;
     END
    ELSE IF I < 0 THEN ! REAL GREATER THAN DESIG.
                       ! WE HAVE PASSED
     BEGIN             ! LAST GOOD ENTRY
         RECIP'RET.(13:2) := 2; ! STOP SCAN
         GO TO GET'LOST;
     END;
    GETUSERMODE;

<<   NOW, FORMAT THE LINE AND PRINT IT.   THIS IS WHERE THE
CUSTOMIZATION WILL GO IF THIS PROGRAM IS TO BE MODIFIED
>>

    CLEARBUFF;
```


Copyright (C) 1989 Propellerhead Publishing

Directory Expansion(continued)

```
    IF  LEVEL = FILE'LEVEL THEN
        MOVE BBUFF := "FILE ",2
    ELSE IF LEVEL = GROUP'LEVEL THEN
        MOVE BBUFF(4) := "GROUP ",2
    ELSE IF LEVEL = ACCOUNT'LEVEL THEN
        MOVE BBUFF(8) := "ACCOUNT ",2
    ELSE IF LEVEL = USER'LEVEL THEN
        MOVE BBUFF(12) := "USER ",2
    ELSE IF LEVEL = VOLUME'LEVEL THEN
        MOVE BBUFF(16) := "VOLUME ",2;

    MOVE * := DL'POINT'BA,(8),2; !MOVE NAME OF ENTRY
    LEN := TOS - @BBUFF;

<<  FOR THE FILE ENTRY, PRINT THE DISC ADDRESS ALSO
>>
    IF LEVEL = FILE'LEVEL THEN
      BEGIN  << PUT COMMA AT END OF NAME >>
        MOVE BBUFF(LEN-8+SCAN BBUFF(LEN-8)
                UNTIL "  ") := ",";
        LEN := LEN + 1;
        LEN := LEN + MOVE BBUFF(LEN) := "LDEV = ";
        LEN := LEN + ASCII(DL'POINT'LA(4).(0:8),
                           10,BBUFF(LEN));
        DL'POINT'LA(4).(0:8) := 0;
        LEN := LEN + MOVE BBUFF(LEN) :=
                ", ADDRESS = %";
        LEN := LEN + DASCII(DL'POINT'DA(2),
                            10,BBUFF(LEN));
      END;

<<  NOW THAT WE HAVE FORMATTED THE ENTRY, LET'S PRINT IT
>>


    PRINT(LBUFF,-LEN,0);
GET'LOST:
<<  OKAY, NOW WE HAVE TO SWEEP UP AFTER OURSELVES, AND EXIT BY
TELLING DIRECSCAN WHAT TO DO
>>
     GETPRIVMODE;
     SETCRITICAL;   << TURN ON CRITICAL BIT >>
     PROCESS'AN'ENTRY := RECIP'RET;
     EXCHANGEDB(DDSDST);
END;  << OF PROCESS'AN'ENTRY >>

BEGIN   << OF MAINLINE >>

  GETUSERMODE;
  PUSH(STATUS);
  TOS.(2:1) := 0;
  SET(STATUS);  <<  TURN OF TRAPS >>
```

Copyright (C) 1989 Propellerhead Publishing

Directory Expansion(continued)

```
  DISPLAY "SCANDEMO (C) MATTEDOR " NOCRLF;
  DISPLAY "COMPUTER SERVICES, 1988" ENDDISP;
  DISPLAY "" ENDDISP;

GET'FILES:
  CLEARBUFF;

  DISPLAY "PLEASE ENTER FILESET TO EXAMINE -->"              NOCRLF;
  CLEARBUFF;
  LEN := READX(LBUFF,-80);
  IF < OR LEN = 0 THEN GO TO EXIT';   << IF EOF OR
                   NOTHING KEYED IN >>

  GETPRIVMODE;
  FLAG := PRODUCEPARMS(PARM,BBUFF,
         LA'RESULT,DELIM,ERROR);
  IF NOT FLAG THEN BEGIN
     GENMSGU(2,ERROR);
     GETUSERMODE;
     GO TO GET'FILES;
     END;

  PRINT(LBUFF,0,0);  << PRINT BLANK LINE >>
  DRET := DIRECSCAN(PP'TYPE,PP'INDEX,PP'D'LA'ACCTNAME,
                    PP'D'LA'GROUPNAME,PP'D'LA'FILENAME,
                    PROCESS'AN'ENTRY,
                    LA'RESULT,0); ! 0 = SYSTEM DOMAIN
   IF <> THEN
     BEGIN
       IF > AND DRET2 = 2 THEN  ! DIDN'T FIND A GROUP
         BEGIN
          CLEARBUFF;
          MOVE BBUFF := "INVALID ",2;
          CASE DRET1 OF BEGIN
            MOVE * := "FILE ",2;
            MOVE * := "GROUP ",2;
            MOVE * := "ACCOUNT ",2;
            MOVE * := "USER ",2;
            MOVE * := "VOLUME SET ",2;
          END; << OF CASE >>
          MOVE * := "NAME SPECIFIED",2;
          LEN := TOS - @BBUFF;
          PRINT(LBUFF,-LEN,0);
         END
       ELSE BEGIN
         PUSH(STATUS); << SAVE CC FOR CHECK >>
         RESETCRITICAL(0);
         SET(STATUS);
         IF < THEN DISPLAY "CCL RETURNED BY DIRECSCAN"
             ENDDISP;
         IF > THEN DISPLAY "CCG RETURNED BY DIRECSCAN"
             ENDDISP;
         CLEARBUFF;
         LEN := MOVE BBUFF := "DRET1 = ";
```


Copyright (C) 1989 Propellerhead Publishing

Directory Expansion(continued)

```
          LEN := LEN + ASCII(DRET1,10,BBUFF(LEN));
          LEN := LEN + MOVE BBUFF(LEN) := ",DRET2 = ";
          LEN := LEN + ASCII(DRET2,10,BBUFF(LEN));
          PRINT(LBUFF,-LEN,0);
          QUIT(100);
        END;
      END;
  PRINT(LBUFF,0,0);
  GO TO GET'FILES;
EXIT':
END;
END.
```

If you examine the PROCESS'AN'ENTRY procedure carefully, you will see the preamble and postamble code that deals with the environment that DIRECSCAN set up. The mainline calls PRODUCEPARMS to format the parameters correctly. The entry type (*leaf* parameter of PRODUCEPARMS) is taken directly from the PARM that the program is run with. Thus, a type of zero (the default run PARM) will indicate to PRODUCEPARMS that files are desired. By running the program with different PARMS, different entries will be visited.

While PROCESS'AN'ENTRY does not really do much but display the entry names as they are visited, the manner in which the names are checked is very important. DIRECSCAN does not perform generic file name matching automatically. It really only understands visiting specific accounts, groups, or files, or visiting all of an entity. If a generic name is specified, then RECIP (PROCESS'AN'ENTRY in this case) must choose to process the entry or not. In some cases, RECIP might decide that no more entries can possibly be visited in the scan. This is possible, since all the entries are presented to RECIP in sorted order.

For example, if files in the set @.@.S@ are being scanned, and the account entry for TEST is presented to RECIP, this has to be beyond the file set being scanned. Rather than having DIRECSCAN continue with the scan and RECIP just ignore all the entries, it is possible for RECIP, via its integer return value, to communicate with DIRECSCAN. Two bit fields are detected by DIRECSCAN. `RECIP.(15:1)` informs DIRECSCAN whether or not the directory SIR was released (this will normally be yes, or set to zero). `RECIP.(13:2)` instructs DIRECSCAN in how to proceed. A value of zero in this field tells DIRECSCAN to continue the traversal. A value of 1 indicates that the current subtree is to be skipped. This is only useful if RECIP is processing an entry with a subtree, namely a group or account. If file entries are being visited, then all the entries for the current group must be processed. However, if we are visiting a group which does not match the group name criteria of the search, a value of 1 returned will cause DIRECSCAN to skip to the next group (subtree). The same goes for accounts.

The easiest way to perform the pattern matching is with the procedure DIRMATCH (page 2-97). The procedure PROCESS'AN'ENTRY above uses it to check each name as it is presented. The LEVEL parameter is checked, and the proper adjustment is made. DIRMATCH returns either -1, 0, or 1 to indicate that the generic name is less than (alphabetically), matches, or is greater than the actual entry name provided, which is the entry name from the NTRY parameter. If the generic (designator) is less than the real name, then we have passed it in the scan, and we can stop the scan completely if we are on an account entry, otherwise all we can do is stop the scan of our current **parent's** subtree. For example, if we scanning the file set p@.u@.@ and encounter the file QUERY.USER.SYS, then passing a 2 back in this subfield will not stop the scan totally, but will get the next group in the account we are visiting.

To summarize the returns for RECIP, `RECIP.(15:1)` instructs DIRECSCAN whether the SIR was released. `RECIP.(13:2)` can return 3 values, defined as follows:

Copyright (C) 1989 Propellerhead Publishing

0 - Continue to the next appropriate entry
1 - Skip current subtree. "Visit the next thing of the same level as me, unless at the end of the entries at my level."
2 - Stop scan of parent's subtree. "Don't visit any more of the same level as me. Return to next level up, get the next entry, and continue."

Notice that there **is no way to totally stop the scan from a lower level.** If a file entry is being visited, we cannot stop the scan totally. This can only be done when visiting an account entry. Since there is no level above, this will terminate the scan totally. If it is desirable for the RECIP procedure to stop the scan totally from any level, then a flag must be set (in a global area or PARMS), then checked at the beginning of the RECIP procedure. If set, the procedure must exit immediately setting this subfield of the integer return field to 2.

Conclusion

The directory procedures are extremely useful. Some very straightforward to use, some not. With careful use and understanding of these procedures, some very useful functions can be added to programs.

Copyright (C) 1989 Propellerhead Publishing

IO System

Easily one of the most powerful facets of MPE is the ability to communicate with IO devices. Since a computer systems ultimate usefulness depends upon communication with terminals, discs, printers, etc., the ability to manipulate these devices is critical.

MPE provides many high level facilities for communicating with devices, most notably the File System. As with any other high level facility, the File System insulates the programmer from worrying about a great number of IO device idiosyncracies. However, it also isolates the programmer from being able to perform certain very powerful functions. Reading specific areas of disc, printer carriage control, setting terminal status functions, and communicating with devices owned by other processes are just a few of the things that can be accomplished with the IO system.

ATTACHIO

The main interface into the IO system is the procedure ATTACHIO. ATTACHIO's purpose in life is to take the parameters passed to it, format them into an Input-Output Queue (IOQ) or Disc Request Queue (DRQ) entry, then attach this entry to a linked list of entries for that particular LDEV. It then awakens the IO system, and the IO is initiated. The lower level IO system takes over, and the IO driver for that particular device picks up the IOQ/DRQ entry, finds out how the IO is supposed to be performed. After the IO is complete, the process is re-awakened (if the proper bit in the *flags* parameter is set indicating a blocked IO) and the process continues.

ATTACHIO, then, really is just a messenger. It really doesn't do very much, other than format an entry and put it in a table. The secret to using ATTACHIO is to understand the data that must be placed in this "IO driver message", and from this what data to pass in ATTACHIO's 9 parameters. This section will outline the main parameters.

ATTACHIO Parameters

The parameters for ATTACHIO are defined on page 2-18. The following is just a verbal description of the parms, given our new point-of-view that we are just formatting a driver message.

Ldev is just that...the ldev number of the device we are performing the IO on. Care must be taken to pass a valid ldev number, as an invalid one can cause a system failure 206.

Qmisc is a general, miscellaneous parameter that is defined by the driver. For virtually all calls, this is set to zero. It is really just a way of passing another word of device specific data to the driver.

DSTX is the DST number where the data to be written or the buffer to read into exists. This is very powerful feature of ATTACHIO, as it gives the programmer the ability to perform an IO into an external data buffer. If a value of zero is passed, then the calling process's stack is assumed to be the location of the data buffer. One of the things that ATTACHIO does for us is to put the calling process's pin number into the IOQ or DRQ entry, so it is fairly easy to match the pin and stack dst number.

Addr is the address of the data buffer, which varies depending upon the value of *dstx*. If *dstx* is zero, then this is a DB relative address for the data buffer. If *dstx* is non-zero, then this address is relative to the beginning of the data segment.

Func is a very device dependent function code. The ATTACHIO definition on page 2-18 lists the most common ones, and virtually all IO drivers conform to these common codes. However, each driver defines

Copyright (C) 1989 Propellerhead Publishing

its own set of functions. Some, such as the DS pseudo device drivers and some others, have a multitude of functions they support. The tables that follow define a few function codes.

Count is the number of bytes or words to transfer, or are expected to transfer. It follows the standard MPE conventions. Negative numbers are bytes, positive words.

P1 and *P2* are again very device dependent parameters. There are no standard values for these, and are defined by the individual driver. The tables that follow document some of these values.

Flags is a critical parameter that must be set correctly in order for ATTACHIO to function correctly. A few definitions are in order here. Please refer to page 2-18 and the definition of *flags*.

The preemption bits are used to determine where in the queue for that ldev ATTACHIO should place this entry. If this field is set to zero, then no preemption is done and the entry is queued at the end of the chain for this ldev. Soft preemption will cause ATTACHIO to queue the IOQ entry after the first entry, if the IO is in progress, and before the first entry if the IO has not started. The :TELL command uses soft preemption. Hard preemption causes the current IO to be aborted and the IO being queued to be executed. An operator :WARN uses this option.

Diagnostic requests are usually set by system code when requesting status from a device or some similar function. This bit should not normally be set.

The system buffer flag, combined with correct request type bits, allows data to be placed in a system buffer (a defined DST) and the request queued, then have the process terminate with the IO completing when it should. This used to be used by :TELL's and :WARN's. If a read is in progress on the terminal that is the target of the :TELL, the IO will not complete until the read is complete. Rather than tie up the process while the read completes, the IO is queued from an SBUF. When the IO completes, the SBUF is deallocated.

The request type combines several options: block[4] or noblock, impede if no IOQ available, no PCB associated with this request. The "normal" value is a 1 (blocked, impede on no IOQ available). A value of zero will allow the process to continue, doing nowait IO. In this case, the return from ATTACHIO is the IOQ index, necessary to later check on the status of the IO with IOSTATUS (page 2-173). It is up to the application to perform this checking. One of the powerful things that direct calling of the IO system allows is the caller can manage multiple nowait IO requests simultaneously. The file system, when using nowait, allows only one IO in progress for a file. This is because the file system only has room to store on IOQX number per file.

IOCDPN0

A utility program exists in PUB.SYS called IOCDPN0. It is a generic driver program that allows a call to ATTACHIO to be formatted and executed, analogous to DBDRIVER for TurboIMAGE. This Expansion will not discuss it here, but a fairly good help facility exists. When you run it, type "X" at the prompt (short for XPLAIN). **Do not type HELP or "H".** This calls the low level debugging procedure HELP, and locks the system up. If this happens, type an "R" on the console (resume).

Use the commands documented by X to enter the ldev, function, qmisc, count, p1, p2, and flags. The buffer is normally an internal one to IOCDPN0, but an external dst can be specified. Then type E for execute, and ATTACHIO is called. If you are careful and have entered the parameters correctly, the ATTACHIO should execute correctly.

---

[4]. The term block in this context does not relate to a "block" as defined by the file system. It refers to the fact that a process is impeded, or "blocked" from proceeding while the IO takes place.

Copyright (C) 1989 Propellerhead Publishing

IO Expansion(continued)

Charts

The following chart go beyond the definitions on page 2-18 in some ways. The data for the functional return of ATTACHIO is listed, along with the P1/P2 and function codes.

Functional Returns

Qualifying Status(8:5)	General(13:3)	Overall(8:8)
1 - Waiting for Completion	0 - Pending	1
2 - Doing Error Recovery	2	2
3 - Not Ready Wait		3
4 - No Write Ring Wait		4
5 - No Paper Tape Wait		5
0 - Normal	1 - Successful	1
1 - Read Terminated with Spec Char		%11
2 - Tape Retry Required		%21
3 - Low tape or EOT		%31
1 - Physical End of File	2 - End of File	%12
2 - :DATA		%22
3 - :EOD		%32
4 - :HELLO		%42
5 - :BYE		%52
6 - :JOB		%62
7 - :EOJ		%72
1 - Terminal Parity Error	3 - Unusual Condition	%13
2 - Terminal Read Timed Out		%23
3 - IO Aborted Externally		%33
4 - Data Lost		%43
5 - DSR Dropped or Unit Not Online		%53
6 - Aborted For Power Fail		%63
7 - BOT and DSR, BSF Request		%73
%10 - Tape Runaway		%103
%11 - EOT and Write Request		%113
%12 - No Write Ring After OPReply		%123
%13 - End of Tape		%133
%14 - Plotter Limit Switch Reached		%143
%15 - Enable Subsys Break and No CY Pin		%153
%16 - Read Time Returned Overflow		%163
%20 - Write and No Card in Wait Station		%203
0 - Invalid Request	4 - Irrecoverable Error	4
1 - Transmission Error		%14
2 - I/O Time Out		%24
3 - Timing Error		%34
4 - SIO Failure		%44
5 - Unit Failure		%54
6 - Invalid Disc Address		%64
7 - Tape Parity Error		%74

Copyright (C) 1989 Propellerhead Publishing

IO Expansion(continued)

%11 - Paper Tape Tape Error	%114
%12 - System Error	%124
%13 - Invalid SBUF Index	%134

Disc Operations

Operation	Funct	P1	P2	XLOG	Comments
Read	0	DBL	DiscAddr	-	Odd byte count always rounded
Write	1	DBL	DiscAddr	-	Cnt rounded up to mult of 128
File Close	3	-	-	-	
Device Close	4	-	-	-	
Zero Fill	5	DBL	DiscAddr	-	Filled with %000000
Space Fill	6	DBL	DiscAddr	-	Filled with %020040
Request Status	7	-	-	-	
Format Track	8	DBL	DiscAddr	-	
Initialize Track	9	DBL	DiscAddr	-	
Read Full Sector	10	DBL	DiscAddr	-	138 Words/Sector
Write Label	11	DBL	DiscAddr	-	Sector 0
RD W/O Sparing	12	DBL	DiscAddr	-	

Mag Tape

Operation	Funct	P1	P2	XLOG	Comments
Read	0	EOF	-	XCNT	
Write	1	-	EOT Flag	-	
File Open	3	-	-	-	NOP
File Close	4	-	-	-	NOP
Device Close	4	-	-	-	Rewind and Unload
Rewind	5	-	-	-	
Write EOF	6	-	-	-	
Forward Space File	7	-	-	-	
Backward Space File	8	-	-	-	
Rewind and Unload	9	-	-	-	
Gap Forward	10	-	-	-	
Forward Space Record	11	-	-	-	
Backward Space Record	12	-	-	-	

Line Printer

Operation	Funct	P1	P2	XLOG	Comments
Write	1	VFC	MODE	-	
File Open	2	-	-	-	TOF if not there already
File Close	3	-	-	-	TOF if not there already
Device Close	4	-	-	-	TOF if not there already
Download VFC	64	6/8LPI		-	CNT, ADDR&DSTX=VFC Buffer
Set Left Margin	65	Mrgn	Flag	-	if P2.(15) set, store P1 new default margin

Terminals

Copyright (C) 1989 Propellerhead Publishing

IO Expansion(continued)

Operation	Funct	P1	P2	XLOG	Comments
Read	0	EOF	Stop Char	-	P1.(0;1) = 1 NO LF on P1.(13:3) = EOF SPEC P2.(0:8) = IF<>0 Spec P2.(11:2) - 0/<>0 Ascii
Write	1	Ctl	Flag	-	P1 = 0 Insert cr/lf P1 = 1 Use byte 1 of data P2.(11:2) = if<> binary P2.(15:1) - Prespace
File Open	2	-	-	-	NOP
File Close	3	-	-	-	Dumps partial rec & CR/LF
Device Close	4	-	-	-	Disconnect if Modem Hang
Set Timeout	5	#secs	-	-	P1 = 0 to clear timeout
Set In Speed		6	Chrs/sec	-	Oldspeed Returned XLOG is old speed
Set Out Speed	7		Chrs/sec	-	Oldspeed Returned XLOG is old speed
Set Echo On	8	-	-	Oldstate	Returned XLOG 1/0 = on/off
Set Echo Off	9	-	-	Oldstate	Returned XLOG 1/0 = on/off
Disable Break	10	-	-	-	
Enable Break	11	-	-	-	
Disable CY	12	-	-	-	
Enable CY	13	-	-	-	
Disable PTAPE	14	-	-	-	
Enable PTAPE	15	-	-	-	
Disable Timer	16	-	-	-	
Enable Timer	17	-	-	-	
Read Timer	18	-	-	sec/100	XLOG in 1/100 Secs
Disable Parity	19	-	-	-	
Enable Parity	20	-	-	-	
Logged On	21	Type	-	-	P1 = 1 for session
Not used	22	-	-	-	
Set Termtype	23	TTYpe	-	-	P1 = MPE Term Type
Allocate Term chars/sec	24	TTYPE	Speed	-	P2 = Speed
Clear flush and Write	25	Cntrl	-	-	Clr Partial Record
CTRL-X Echo On	26	-	-	-	
CTRL-X Echo Off	27	-	-	-	
Invalid Request	28				
PTAPE READ	29	DBL	DiscAddr	-	Address of Spoolfile

Copyright (C) 1989 Propellerhead Publishing

Set Break Mode odd/even=set/clr	30	on/off	-		P1
Set Console Mode odd/even=set/clr	31	on/off	-		P1
Set Parity	32	Parity	-		P1 = 0 - None 8 bit P1 = 1 - None 8 bit P1 = 3 - Odd Gen parity P1 = 4 - Even Gen parity
Allocate Term	33	TType	Speed	-	
Set Term Type	34	TType	-	-	
Get Term Type	35	-	-	Termtype	
Get Out Speed	36	-	-	Outspeed	
Set Stop Chars	37	Stop	-	-	P1.(0:8)=Subsys break P1.(8:8)=EOR P1 = 0 Disable
Set Console Interrupt	38	on/off	-	-	P1 odd = Enable CTRL A P1 even = Disable CTRL A

Conclusion

The data passed to ATTACHIO is really just a message to the driver. For this reason, the ultimate authority on what is correct to pass to ATTACHIO is the driver code itself. If you are interested in a much more detailed level of information for a specific device, you should acquire the MPE-V source and get the driver source for that device.

Copyright (C) 1989 Propellerhead Publishing

MPE-V Security

Procedures: CRYPT, GETSALT, GETSECINFO, SECHK'PASSWORD, SECURITYINSTALLED, VERIFY'PASSWORD

Description

In response to the growing concern in computer security, Hewlett-Packard introduced a new product focused on enhancing the security of HP computers for their security conscious customers. To achieve these additional security features several new procedures and another internal table were introduced. However, scant information was released (understandably so) on this new facility in MPE-V.

New Security Procedures

To facility the new security features, six procedures were added to MPE-V. The procedures themselves are fairly straightforward and easy to use. However, there is one major consideration which must be taken into account when attempting to access several of these procedures. The procedures CRYPT, GETSALT and SECURITYINSTALLED were written in PASCAL and therefore have PASCAL parameter checking. This makes it impossible to call them from any languages other than PASCAL or SPL because the parameter checking flags for PASCAL are not compatible with FORTRAN-77 or COBOL-II. And the only reason they can be called from SPL is because in SPL the programmer can control the parameter checking level and can specify 'no checking'.

Locating Commands

In order to locate a command in the command table (Table 3) it is necessary to obtain the system directory number associated with the command. This is done by using the COMSEARCH procedure, documented on page 2-56.

Device Password Encryption

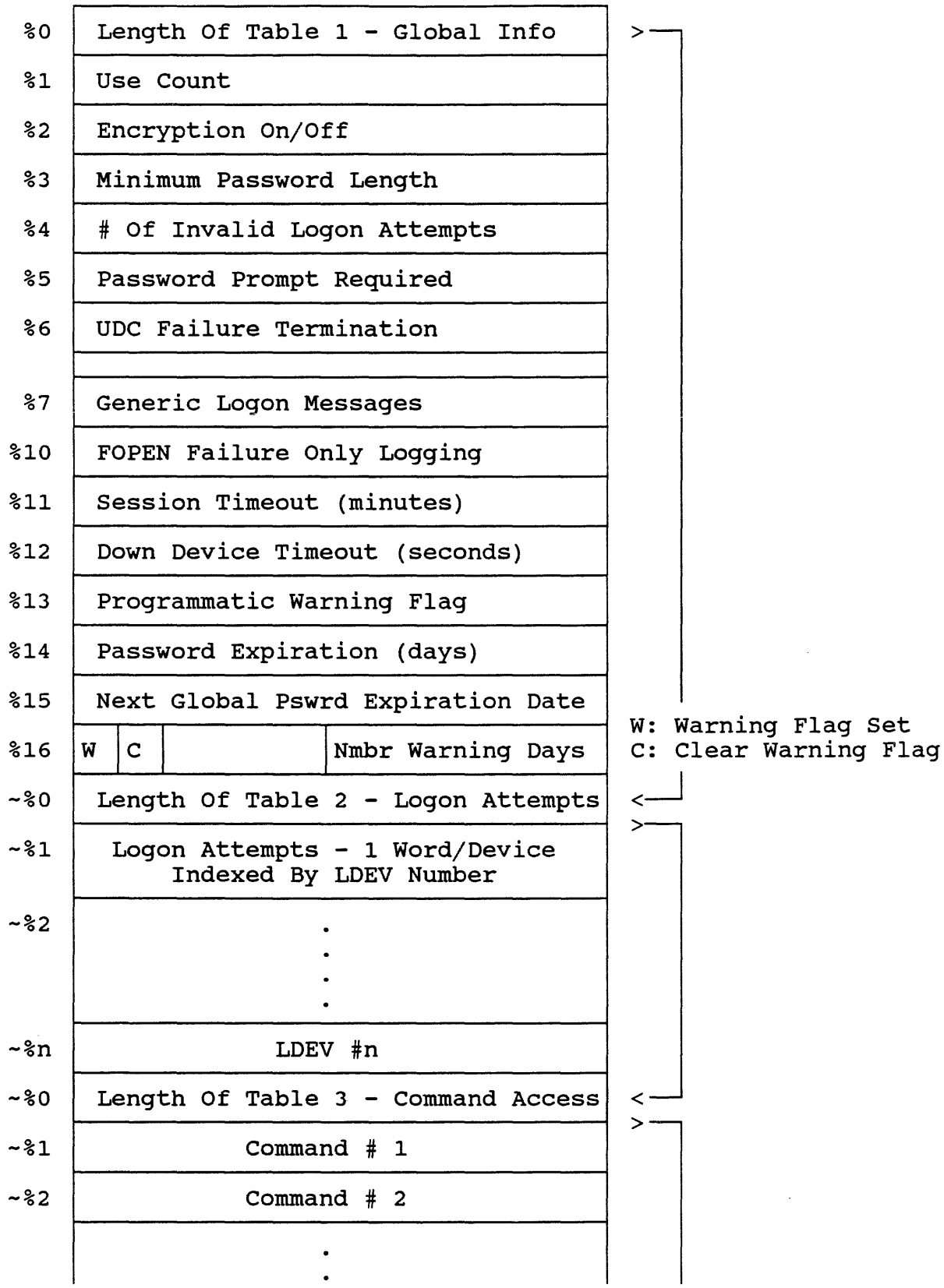
Passwords associated with a device in the table are stored in a one-way encryption. It is may be possible for one to decrypt them given the resources of the National Security Agency but this is not very practical. The easiest way to determine if a password matches the one in the table is to encrypt the supplied password and compare the one in the table. In order to do this one has to call the CRYPT procedure using a 'seed' containing the word *DEVICEPW.*

System Security Table

To maintain information on the configuration of specific security features HP added a new table to the multitude of those which already exist. Since this table is not documented anywhere, we felt that this would be a good thing to include here. This new table is identified by a DST number maintained in word %30 of the System Global Extension area. The format of the table is as follows:

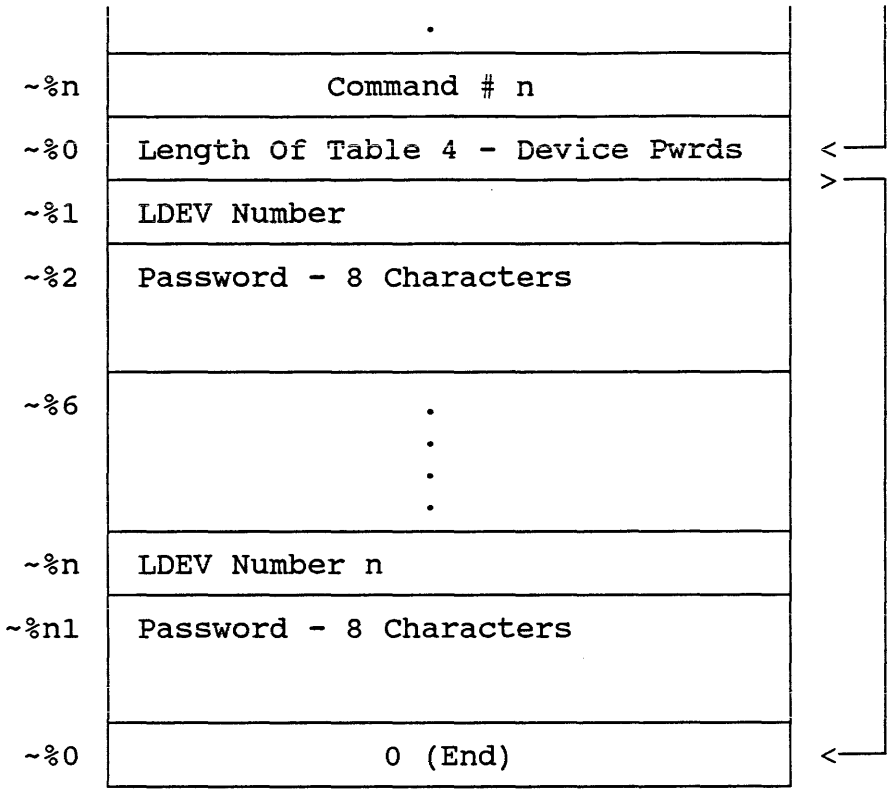
Copyright (C) 1989 Propellerhead Publishing

Security Expansion(continued)



Copyright (C) 1989 Propellerhead Publishing

Security Expansion(continued)



Command Access Entry

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
P	E	L													

P - Programmatic Access Disabled
E - Command Disables
L - Command Logged

The security information is maintained in both the MPE-V table and also in the file SECDATA.PUB.SYS. The file copy is used to load the memory table during initialization.

Copyright (C) 1989 Propellerhead Publishing

About the Authors

Jerry Fochtman

After graduation from Western Michigan University in 1974, Jerry Fochtman went to work for the Kalamazoo Public School System. In October of that year, the school district took delivery of their first HP-3000 computer system (Model CX). With his first logon began Mr. Fochtman's long affiliation with Hewlett-Packard computers. During the ensuing years Mr. Fochtman has worked on almost every model of HP-3000 which has been produced.

In 1980 Mr. Fochtman moved to Houston Texas as a consultant to Exxon Chemical Company. In that capacity he was responsible for the development of several 'real-time' type of applications on the HP-3000 which interfaced to process control systems. He was also involved in the development of several computer performance tools along with utilities to enhance HP's computer security. In 1984 he joined Exxon as an employee.

Mr. Fochtman started sharing his HP experiences with an article published in the first issue of the Hewlett-Packard Journal (Vol 1, No. 1, May, 1977). Since that time he has written numerous papers for local publications and occasionally in some of the national and international publications for Hewlett-Packard users. He has also made several presentations at both the local and the national levels.

In addition to his presentations and papers, Mr. Fochtman has provided contributed many utilities to the INTEREX Contributed Library. Several of which have been adopted by various organizations throughout the world.

Mr. Fochtman has been on the Board of Directors for the Greater Houston Regional Users' Group since 1984 and has served as Board Chairman since 1986.

Jason M. Goertz

Jason Goertz graduated from Whitman College in Walla Walla, Washington in 1978, where he stayed on to work for the Computer Center. During his time there, he worked on the early versions of the Contributed Software Library, and donated many programs to the CSL.

In 1980, Mr. Goertz went to Hewlett-Packard where he spent 4 1/2 years as a System Engineer and System Specialist, with emphasis on MPE Internals, Performance, and served as a Field Software Coordinator. An experienced instructor, he taught advanced topics to HP customers, such as the SE335 class (MPE Internals) and SPL. He also contributed several utilities to the HP TELESUP account that are still in use today.

In 1984, he left Hewlett-Packard and assumed the duties of Vice President of Information Systems for Generra Sportswear, a Seattle based garment importer. In 1985, Mr. Goertz started his own consulting company, Mattedor Computer Services, specializing in HP3000 and microcomputer service based products.

He is periodically retained by the HP offices on the west coast to teach the MPE Internals Class, and is consulted extensively in 1986 and 1987 for HP's Integrated Technology Group working on The Spectrum Project, first testing MPE-XL, then on TurboImage/XL. Currently, Mr Goertz is retained by HP in California to teach MPE-XL internals.

Mr. Goertz is an author of the SPLash! native mode SPL compiler, in addition to being a contributor to the book *Beyond Risc!* He has written many papers for user group meetings, as well as articles for HP publications. He is active in the Northwest Regional User's Group (NOWRUG). He currently has a regular column in the HP Chronicle, Notes from Swan Lake. Mr. Goertz makes his home in Sumner, Washington.

FOLD FOLD

25¢
Postage

Propellerhead Publishing Group
5508 183rd Avenue East
Sumner, Washington
98390

Attn: Technical Research Dept.

FOLD FOLD

Index

:ALTACCT 1
:LISTF 1, 12
:NEWACCT 1, 8
:REPORT 1, 12
:TELL 24
:WARN 24
$CONTROL PRIVILEGED 3
ABORT 1
ABORTIO 4
ABORTJOB 5
ABORTPROCIO 6
ABORTTIMEREQ 7
ACCCHECK 8
ADDJTENTRY 10
ADOPT 12
ALLFLAG 12
ALLORIN 13, 60
ALTDSEGSIZE 15
ALTEROUTPUTFILE 16
ATTACHIO 1, 18, 74
AWAKE 22
BUILDSTACKMARKER 24
BURRYPROC 25
C'CLOCK 26
C'DATE 27
CACHE'LDEV 28, 251
CDT'SET'POST 29
CDT'SET'RND 30
CDT'SET'SEQ 31
CENTER 32
CHANGEDB 33
CHECK'BOT 35
CHECKALIVE 36
CHECKALLOW 37
CHECKDISC 39
CHECKLDEV 41
CHEK 42
CLEANUPFILES 45
CLEAR'BREAKPNT 49
CLEAR'PSIF 46
CLEARWWS 48
COBOL-II 29
COMSEARCH 29, 56
CONSALTFILE 52
CONVERTDATE 53
CONVERTTIME 54
CONVEXTLABELTODELTAP 55
CRYPT 29, 58
DATEJUL 59
DEALLORIN 60
DEBLANK 61
DELAY 62
DELETEDEVFILE 63
DELINKSCHED 65
DEVSPEC 66
DIRECADJUST 2, 67
DIRECFIND 2, 4, 5, 69
DIRECFINDFILE 2, 4, 5, 7, 72
DIRECINSERT 9, 75, 80
DIRECINSERTFILE 10, 68, 77, 78
DIRECLOGOFF 81
DIRECLOGON 83
DIRECPURGE 6, 7, 85, 89
DIRECPURGEFILE 6, 7, 10, 68, 87, 88
DIRECRESETFILE 80
DIRECSCAN 2, 11-13, 90, 101, 194
DIREC[RE]SETFLAG 94
DIRESCAN 12
DIRMATCH 21, 97
DISCSIZE 98
DISCTYPE 99
DISMOUNT 100, 101, 194
DSMOVE 102
DSOPEN 103
DST'SIZE 104
ENDLEVEL 11
ERROREXIT 105
ERRORON 13, 107
EXCHANGEDB 12, 108
FALTSEC 109
FAMILY 111
FGETLOCKWORD 112
FINDDEVICES 113
FLABIO 7, 74, 115
FLUTIL 6
FMAKETEMP 117
FORMSG 118, 121, 134
FORMUSERID 122
FORTRAN 29
FREEZESEG' 123
FREPLY 125
FROMCALENDAR 126
FSCLOSE 127
FSECTORS 7, 128
FSOPEN 129
GENMSG 121, 131
GENMSGU 135
GET'DSDEVICE 137
GET'FILECODE 139
GET'FILEMNEMONI 140
GET'SYS'PRIV 141
GETCLASS 142
GETCLOCK 144
GETDATASEG 146

GETDEVINFO 148
GETJITDST 150
GETJOBN 151
GETMAXJS 152
GETPCBXMEASINFO 153
GETPRIVMODE 3
GETSALT 29, 155
GETSECINFO 157
GETSIR 13, 158
GETSTATISTICS 161
GETUSERMODE 3
GIVECLOCK 163
HELPPROC 164
HITFLAG 12
IDENTIFY'TERMIN 165
IMPEDE 167
IO system 1
IOCONTROL 168
IOFREEZE 169, 171
IOSTAT 173
IOUNFREEZE 175, 177
ISSYSCACHABLE 179
JULDATE 180
LDEVTOTYPE 181
LDEVTOVTAB 182
LOCKJIR 183
LOCKSEG 184, 186
LOGICALCST 188
LUN 189
MAKEFN 190
MASTEROP 191
MAX'DST'SIZE 192
MOUNT 3, 6, 74, 100, 193
MPE 195
MUSTOPEN 196
MYPIN 197
OBTAIN 198
PASCAL 29
PCB'INDEX 200
PHYSICALCST 201
PINJOBMAP 203
Private Volumes 3
PROCERSS'CODE'LOAD'INFO 204
PROCFILE 206
PRODUCEPARMS 12, 21, 207
QUEUEPROC 209
RECIP procedure 11, 12
RELDATASEG 210
RELEASE 211
RELSIR 13, 212
REMOVE'PASSWORDS 213
REQUESTSERVICE 214
RESETBREAKBITS 215
RESETCRITICAL 13, 216
RESETDB 217
RLOCK 218, 219
RUNLOCK 219
SCHEDDAYTIME 220
SCHEDULESCHED 221
SECHK'PASSWORD 222
SECURITYINSTALLED 29, 224
SEGMENTER 225
SET'BREAKPNT 229
SET'PSIF 231
SETCRITICAL 12, 13, 232
SETDBTOSTACK 233
SETSYSDB 234
SFINDODD 235
SHUTDOWNCACHING 236
STACK'DST 237
STACKCHECK 238
STARTLEVEL 12
STARTSTATISTICS 239-241
STOPSTATISTICS 241
STRINGPINATTAIL 242
SYSGLOB 243
SYSPROC 244
Tables Manual 2
TELESUP account 6
THISCPU 245
TIMEREQ 247
TOCALENDAR 249
TOLEVEL 11
UDC 250
UNFREEZESEG 252
UNIMPEDE 254
UNLOCKJIR 255
UNLOCKSEG 256, 258
UNSTRINGHEADPIN 260
VERIFY'PASSWORD 261
VTABINX 263
VTABTOLDEV 264
WAIT 265
WHERES'DB 267
XCODETRAP 268
XRETJTENTRY 269

Reader Comment Form

COMPASS - Low Level MPE-V Procedures
September, 1989

We welcome your comments and suggestions concerning this publication. Your feedback helps us improve our product allowing us to provide better information for your needs. Please take a few moments to complete and return this comment form. Feel free to attach additional information if needed.

Some coments you might consider include:

- Did the document present the information in an easy to follow format?
- Was the information accurate?
- Did the information meet you needs?
- What areas could be improved?
- What did you like/not like about this publication?

Comments:

If you would like a reply, please include the following information:

Date: _____ Phone: ( ) _____ Ext: _____

Name: _____

Organization: _____

Address: _____

_____

_____

_____

FOLD FOLD

25¢
Postage

Propellerhead Publishing Group
5508 183rd Avenue East
Sumner, Washington
98390

Attn: Technical Research Dept.

FOLD FOLD